Online Resource Procurement and Allocation in a Hybrid Edge-Cloud Computing System

Thinh Quang Dinh, *Student Member, IEEE,* Ben Liang, *Fellow, IEEE*,
Tony Q.S. Quek, *Fellow, IEEE* and Hyundong Shin, *Senior Member, IEEE*

***Abstract*—By acquiring cloud-like capacities at the edge of a network, edge computing is expected to significantly improve user experience. In this paper, we formulate a hybrid edge-cloud computing system where an edge device with limited local resources can rent more from a cloud node and perform resource allocation to serve its users. The resource procurement and allocation decisions depend not only on the cloud's multiple rental options but also on the edge's local processing cost and capacity. We first propose an offline algorithm whose decisions are made with full information of future demand. Then, an online algorithm is proposed where the edge node makes irrevocable decisions in each timeslot without future information of demand. We show that both algorithms have constant performance bounds from the offline optimum. Numerical results acquired with Google cluster-usage traces indicate that the cost of the edge node can be substantially reduced by using the proposed algorithms, up to 80% in comparison with baseline algorithms. We also observe how the cloud's pricing structure and edge's local cost influence the procurement decisions.**

***Index Terms*—Mobile edge computing, resource management, competitive analysis**

I. Introduction

Within the last decade, we have witnessed tremendous growth of data as well as the emergence of the Internet of Things. As a consequence, there is an outburst of digital business that utilizes more and more complex applications with heterogeneous resource requirements. To satisfy the increasing demand of computational power, among contemporary solutions, cloud computing is favored due to its high scalability, accessibility, and availability that come with low storage and computing costs [1]. Cloud providers offer Infrastructure-as-a-Service (IaaS), which is a form of cloud computing that provides instances of virtualized physical resources, generally termed virtual machines (VMs). For example, Amazon EC2 [2] and Microsoft Azure [3] are two such services. There are commonly two pricing options to rent virtual resources: *on-demand* and *reservation* [2], [3]. In the first option, the accounting is purely based on the number of instance-hours used, while in the second one, the users pay a reservation fee in advance, i.e., upfront fee, in exchange for free or discounted resource usage over a certain period. The on-demand rental is often considered a costly option, while the same can be said about the upfront fee in the reservation option if the reserved instances are not used sufficiently often. For organizations or users, it is important to achieve cost effective resource procurement and allocation of cloud computing resources.

Resource procurement and allocation in cloud computing environments have been well-studied [4]–[9]. Many of these works focused on resource allocation with only an on-demand pricing model [4]–[6]. Since reserved resources are effective within a period of time, reservation introduces time-correlation in the decision of both resource procurement and resource allocation. Hence, considering both on-demand and reservation pricing options increases the complexity of the problem. However, leveraging the discount prices offered by the reservation option can lead to substantial cost savings [7]–[9]. In [4]–[9], application owners/organizations were usually assumed to possess no computing or storage capacities.

Since the storage and computation cost has dramatically decreased over the last decade, cloud-like capacities have been moving toward the edge network. There are similarities in concept among Edge Computing (EC) [10]–[15], fog computing [16], and cloudlets [17], where services providers or peer helpers, with their own computational and storage power, can implement applications at near-user servers, namely edge devices. However, the edge devices' capacities are limited in comparison with cloud providers. Therefore, it is necessary to investigate hybrid edge-cloud computing systems, specifically, how the edge's capacity and its local processing cost affect the previously mentioned resource allocation problem over cloud computing environments.

There are very few existing works considering hybrid edge-

Manuscript received May 05, 2019; revised September 22, 2019, November 18, 2019; accepted December 20, 2019. Date of publication xx xx, xxxx. This work was supported in part by the MOE ARF Tier 2 under Grant MOE2015-T2-2-104, the SUTD-ZJU Research Collaboration under Grant SUTD-ZJU/RES/01/2016, the SUTD-ZJU Research Collaboration under Grant SUTD-ZJU/RES/05/2016, the National Research Foundation of Korea Grant funded by the Korea Government (MSIP) under Grant NRF-2019K2A9A2A06024389, and a Discovery Grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada. The associate editor coordinating the review of this paper and approving it for publication was L. Le (*Corresponding author:*).

T. Q. Dinh was with Singapore University of Technology and Design, Singapore 487372. Emails (e-mail: quangthinh_dinh@alumni.sutd.edu.sg).

B. Liang is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, M5S 3G4, Canada. Emails (e-mail: liang@ece.utoronto.ca.).

T. Q. S. Quek is with Singapore University of Technology and Design, Singapore 487372 and also with the Department of Electronics Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do, 17104, Korea (e-mail: tonyquek@sutd.edu.sg).

H. Shin is with the Department of Electronic Engineering, Kyung Hee University, Yongin 17104, South Korea (e-mail: hshin@khu.ac.kr).

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

Digital Object Identifier xxxxxx/TWC.xxxxxx

cloud system, or hierarchical fog-cloud networks, where edge devices/lower-tier cloud nodes with their limited resource capacities need to cooperate with high-tier ones. They usually considered free subscription of IaaS services or a simple cost model (e.g., purely on-demand pricing) [18]–[23], which are either impractical or costly. Hence, it remains an open question how edge server parameters, such as computation capacity and processing cost, as well as the public clouds' pricing options, impact the edge resource allocation decision.

This paper considers a hybrid edge-cloud computing scenario with an edge node and a public cloud node. The edge node has its own VMs. However, because the arriving VM requests can exceed the edge node's capacity, the edge node also rents remote VMs from the cloud and allocates requests to either rented remote VMs or its own VMs. We propose an optimization framework where the edge node performs resource procurement and allocation in order to minimize its long-term operational cost. This scenario allows us to analyze how the edge node's total cost can be improved by its capacity and the cloud node's rental options. We first propose an offline pseudo-polynomial algorithm whose decisions are made with full information of future demand. We then propose an online algorithm where the edge node makes irrevocable decisions without knowing future information. Moreover, the proposed algorithms' performance guarantees are derived.

The contributions of this work are summarized as follows:

- An optimization framework is formulated where the edge node exploits its own resources and the cloud's pricing structure to minimize its long-term operational cost. In the offline setting with full information of future demand, since finding an optimal solution is intractable, we propose a pseudo-polynomial approximation algorithm, which is shown to achieve a 2-approximation ratio.
- We then propose an online algorithm that does not require any information of future demand. A noticeable feature here is that the proposed online strategy makes irrevocable decisions in each time slot. It achieves a constant competitive ratio of $\max\{6, \frac{2p}{\lambda}\}$, where p and λ are two hyper-parameters related to the pricing structure which will be defined later in the paper.
- Through simulations based on Google cluster-usage traces [24], we observe that the edge node can significantly reduce its operational cost when the edge capacity is considered. We also observe the impact of the cloud's pricing structure and edge's processing cost on the procurement decisions.

The rest of the paper is organized as follows. In Section II, we present the related work. Section III describes the system model and the problem formulation. In Section IV, we propose an offline algorithm to solve this problem which has pseudo-polynomial running time. Section V proposes an online algorithm for this problem and presents its performance guarantee. Section VI discusses the empirical evaluations based on real-world traces. Conclusions are then given in Section VII.

II. Related Work

Resource procurement and allocation have been well-studied in many existing works on cloud computing. Some works focused on resource allocation with a simple procurement model (e.g., applying just on-demand pricing) [4]–[6]. Mao and Humphrey [4] proposed heuristic workflow scheduling strategies which minimized the execution cost of the workflow. They tried to ensure the jobs' execution deadline as a soft constraint. The Dynamic Provisioning Dynamic Scheduling algorithm was proposed by Malawski *et al.* [5], which maximized the number of executed workflows under some quality-of-service constraints. In [6] with the same objective as in [4], tasks on a partial critical path were allocated on the same instance by Abrishami *et al.*'s algorithms.

On the other hand, other studies focused on procurement by dealing with multiple pricing options including reserved instances in order to take advantages of discounted prices [7]–[9]. Wang *et al.* [7] considered one on-demand and one reserved instance options and proposed online cloud instance acquisition algorithms without full information of future demand, while Hu *et al.* [8] considered multiple options of different reserved instances in a similar online setting. Hong *et al.* [9] first proposed a dynamic programming method to rent purely on-demand instances to reduce their system's *margin costs*, and then proposed another algorithm utilizing both on-demand and reserved instances to achieve their system's optimal true costs with full information of future demand.

There are few works considering resource allocation in hybrid clouds, edge-cloud, or fog-cloud networks. Existing works usually considered a simple cost model such as free cloud access or purely on-demand pricing [18]–[23]. Chen *et al.* [20], [21] proposed semidefinite-programming based algorithms in order to minimize both energy and latency of workloads in a simple hybrid edge-cloud system. Jiao *et al.* [22] considered a task scheduling problem in multi-tier cloud computing system where the system could jointly optimize its own computational and network resources to reduce the resource allocation cost and resource reconfiguration cost. Furthermore, fog-cloud systems helped to improve the performance of current services by reducing latency and bandwidth consumption in online gaming [18], or the operational cost of medical cyber-physical systems [19]. All of these works considered only on-demand pricing, while ignoring the available discounts through reservation can lead to a costly design. In this work, we leverage the local VMs at the edge node and remote cloud VMs in both on-demand and reservation pricing options.

Our proposed algorithm is an online strategy where the sequence of decisions is irrevocably made without future knowledge [25]. In our problem, the edge node needs to decide whether to reserve instances at any time, which can be classified as a variant of the ski rental problem [26], a class of rent-or-buy problems. The ski rental problem has been expanded in multiple directions such as the Bahncard problem in transportation [27], TCP acknowledgement problem in networking [28], and resource acquisition and resource allocation in cloud computing [7], [8]. In [26]–[28], a decision maker only deals with a single level of demand. The problem in our scenario is more complex as cloud computing demands are in multiple levels (e.g., multiple VMs) [7], [8]. Dealing with multiple levels of demand, Wang *et al.* [7] reduced their prob-

TABLE I
NOTATION USED THROUGHOUT THE PAPER.

Notation	Definition
i, t	index of time
l	index of demand level
d_t	the aggregated demand of arrival VM requests at t
r_t	the number of remote VMs reserved at t
n_t	the numbers of remote reserved VMs that remain active at t
γ	the upfront price for a remote reserved VM
θ	the discount cost for using a remote reserved VM per time slot
p'	the cost to rent an on-demand VM per time slot
λ'	the physical cost of running one VM at the edge per time slot
τ	the reservation period of a remote reserved VM
a_t^r	the number of requests assigned to remote reserved VMs at t
a_t^o	the number of requests assigned to remote on-demand VMs at t
a_t^w	the number of requests assigned to the edge node's VMs at t
w	the number of VMs at the edge

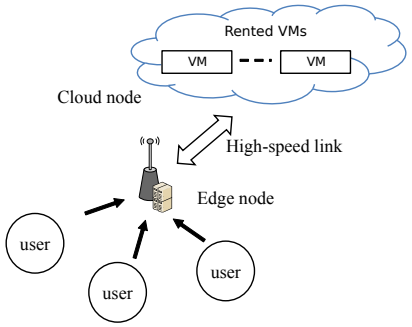


Fig. 1. Computing system with edge and cloud.

lem into multiple independent two-option ski rental problems. Hu *et al.* [8] considered multiple reserved instance acquisition as a two-dimensional parking-permit problem. However, in previous works [7], [8], the number of cloud instances which can be rent in each option is assumed to be infinite. In our work, adding the edge's limited capacity changes the structure of the problem, since here when the edge node's capacity is fully occupied, the excess VM requests will be assigned to the public cloud, in either on-demand VMs or reserved VMs. Hence, our design must account for the impact of the edge's capacity, coupled with the cloud's pricing structure, on the procurement and allocation decisions.

III. System Model and Problem Statement

In this section, we describe the overall system model, explain how the resource at the cloud and the edge is utilized, and state our optimization problem. The commonly used notation throughout this paper is given in Table I.

A. Computing System Model with Edge and Cloud

We consider a multi-tier computing system with one cloud node and one edge node, as shown in Fig. 1. The edge node serves multiple users, such as mobile users or IoT devices, which have computational jobs to be executed. The system is time slotted. User requests arrive at the edge node in every time slot. Let d_t be the total number of VMs requested by users at time t. For simplicity, we assume that each user request lasts for one time slot. However, our system model can be extended to accommodate user requests that last for multiple time slots. In that case, d_t accounts for all VMs from new and on-going user requests in time slot t. However, our analysis neglects the cost of re-assigning these user requests between VMs.

The edge node has its own VMs to process the requests. However, since the capacity of the edge node is limited, the edge node may need to rent remote VMs from the cloud node to scale up its capacity. There are multiple cloud rental options, each of which has a different cost structure. The edge node decides how many remote VMs it should rent and how to assign the arriving VM requests to its own VMs or the rented VMs.

B. Cloud's Resource

The cloud service provider offers the edge node two options to rent its VMs. The first option is called "on-demand" where the edge node can immediately rent a VM that lasts for one time slot with an on-demand price p'. In the second option, called "reserved", if a VM is reserved at time t, it will be effective from t to $t+\tau-1$, where τ is the reservation period. Here, τ is a given value, not a decision variable. Let γ and θ denote the upfront price of renting a single remote reserved VM and the per-slot cost of using a reserved VM, respectively. Obviously, we should have $0 \leq \theta \leq p'$, since otherwise there would be no business case for reserved VMs. Table II shows two examples of on-demand and reservation prices in Amazon EC2. For ease of exposition, we refer to a VM rented with on-demand price as *remote on-demand VM*, and a VM reserved within τ time slots *remote reserved VM*.

Let $r_t \geq 0$ denote the number of new remote reserved VMs that the edge node decides to rent at time t. At time t, the number of remote reserved VMs that remain active is

$$n_t = \sum_{i=t-\tau+1}^{t} r_i. \quad (1)$$

Let a_t^r and a_t^o denote the number of VM requests assigned to remote reserved VMs and remote on-demand VMs, respectively. Clearly, we have $a_t^r \leq n_t$, and $a_t^o = 0$ if there are unused reserved VMs, i.e., $a_t^r < n_t$.

TABLE II
PRICING OF ON-DEMAND AND RESERVED INSTANCES (LINUX, US EAST) IN AMAZON EC2, AS OF JAN. 10, 2018 [2].

Instance type	Pricing option	Upfront	Hourly
m3.medium	On-demand	$0	$0.067
	1-year reservation	$211	$0.016
c3.large	On-demand	$0	$0.105
	1-year reservation	$326	$0.025

C. Edge's Resource

The edge node has its own local capacity w, i.e., the number of local VMs at the edge node. We define a_t^w as the number of VM requests the edge node locally processes and λ' as the cost of locally processing a unit of VM request. The local processing cost is generally defined. As an example, in Section VI, we consider it as the electrical cost incurred by physical processors.

Remark 1. *In the extreme case where the edge node's processing cost is greater than or equal to the price of using a remote on-demand VM, i.e., $\lambda' \geq p'$, we should not use the edge node, and should instead allocate VM requests to remote reserved VMs and remote on-demand VMs. Then, the problem is reduced to the one in [7].*

In the case where the usage cost of a remote reserved VM is greater than or equal to the edge node's processing cost, i.e., $\lambda' \leq \theta$, a trivial solution is to allocate the VM requests to the edge's VMs first. Then, the excess requests are allocated to either remote reserved VMs or remote on-demand VMs, which is again reduced to the problem in [7].

Hence, in this work, we only need to consider the case where $\theta < \lambda' < p'$.

D. Problem Formulation

We consider some time period of system operation T, which is assumed to be a multiple of τ, i.e., $T = K\tau$ where K is a positive integer. The user demands over this time period is $\{d_1, \ldots, d_T\}$. To serve these demands, the edge node decides in each time slot a_t^w, a_t^o, r_t, and a_t^r. Then, its total cost is

$$c = \lambda' \sum_{t=1}^{T} a_t^w + p' \sum_{t=1}^{T} a_t^o + \gamma \sum_{t=1}^{T} r_t + \theta \sum_{t=1}^{T} a_t^r. \quad (2)$$

The first term of (2) is the total cost of processing requests at the edge; the second one is the total cost of using remote on-demand VMs; the third and the final one are the total costs of reserving and using remote reserved VMs, respectively.

Remark 2. *At each timeslot, if $n_t > 0$, the edge node should assign new VM requests to remote reserved VMs first, since $\theta < \lambda' < p'$ as explained in Remark 1. Hence,*

$$a_t^r = \min\{n_t, d_t\}. \quad (3)$$

Furthermore, since $\lambda' < p'$, we should always allocate the remaining requests to local processing at the edge before using remote on-demand instances. Hence, we have

$$a_t^w = \begin{cases} d_t - n_t, & \text{if } 0 < d_t - n_t \leq w, \\ w, & \text{if } d_t - n_t > w, \end{cases} \quad (4)$$

and,

$$a_t^o = (d_t - a_t^w - n_t)^+, \quad (5)$$

where

$$x^+ = \max\{0, x\}.$$

By observing the the relation between n_t, a_t^r, a_t^w and a_t^o as explained in Remark 2, we can rewrite (2) as the following:

$$c = (\lambda' - \theta) \sum_{t=1}^{T} a_t^w + (p' - \theta) \sum_{t=1}^{T} (d_t - a_t^w - n_t)^+ + \gamma \sum_{t=1}^{T} r_t + \theta \sum_{t=1}^{T} d_t, \quad (6)$$

The final term of (6) is the cost of using only pre-reserved VMs to serve all requests, which is the minimum cost to process tasks no matter where they are allocated since $\theta < \lambda' < p'$. The first three terms of (6) are the extra costs if VMs are allocated to other VMs. These terms are analogous to the first three terms of (2).

From the above, we see that the edge node only needs to make a sequence of reservation decisions $\mathbf{r} = \{r_1, \ldots, r_T\}$ to minimize the total cost, i.e.,

$$\mathscr{P}_1 : \min_{\mathbf{r} \in \mathbb{N}^T} \lambda \sum_{t=1}^{T} a_t^w + p \sum_{t=1}^{T} (d_t - a_t^w - n_t)^+ + \gamma \sum_{t=1}^{T} r_t, \quad (7)$$
$$\text{s.t.} \quad (1) \text{ and } (4),$$

where $p := p' - \theta$ and $\lambda := \lambda' - \theta$. Note that since $\theta \sum_{t=1}^{T} d_t$ is a constant, minimizing (7) is equivalent to minimize (6).

We note that $\mathscr{P}_1$ may be viewed as an extension to the cloud instance acquisition problem in [7], where a cloud broker rents remote reserved VMs and remote on-demand VMs to serve users' demand. The cloud broker in [7] can be considered as an edge node without local capacity, i.e., $w = 0$. In our work, since we consider a more general edge node with local computing capacity, its capacity and the local processing cost affect the edge node's cloud instance procurement decisions. This substantially alters the structure of the optimization problem and adds to its difficulty.

In $\mathscr{P}_1$, we focus on the cost at the edge node to serve user demands, without considering the difference in user experience between edge VMs and cloud VMs. However, our formulation is generally applicable. Often the difference in user experience can be negligible, e.g., when the edge node and the cloud are connected by a high-speed link. If it is not negligible, we can modify the cost of edge usage, λ', to reflect the priority of VM utilization due to user experience. However, we note also that a decreasing λ' pushes the problem $\mathscr{P}_1$ toward the second case in Remark 1, and when $\lambda' \leq \theta$, the problem is reduced to the one in [7].

Problem $\mathscr{P}_1$ is combinatorial optimization. It is generally challenging to solve even in the offline setting where the user demands are known in advance. In the more practical online setting, where random user demands arrive dynamically over time, it is even more challenging to design a solution to provide a certain performance guarantee.

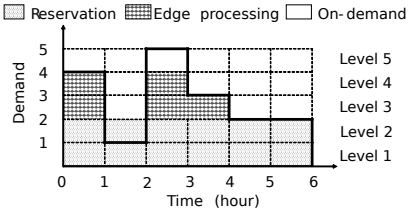


Fig. 2. The resource planning with $\tau = 6$ and $T = 6$, $r = 2$ and $w = 2$.

E. Approximation and Competitive Ratios

In the following, we state the standard definitions of approximation and competitive ratios, which will be used in the evaluation of the performance of the proposed solution.

Definition 1. *Given a sequence of demands* $\mathbf{d} = \{d_1, \ldots, d_T\}$*, let* $c^*(\mathbf{d})$ *denote the offline optimal cost that could be achieved. Suppose an offline algorithm achieves a cost* $c^{\text{Off}}(\mathbf{d})$*. An approximation ratio* ξ *of this offline algorithm is a constant such that for all possible* $\mathbf{d}$*,*

$$\frac{c^{\text{Off}}(\mathbf{d})}{c^*(\mathbf{d})} \leq \xi.$$

Definition 2. *Given a sequence of demands* $\mathbf{d} = \{d_1, \ldots, d_T\}$*, suppose an online algorithm achieves a cost* $c^{\text{On}}(\mathbf{d})$*. An competitive ratio* ζ *of this online algorithm is a constant such that for all possible* $\mathbf{d}$*,*

$$\frac{c^{\text{On}}(\mathbf{d})}{c^*(\mathbf{d})} \leq \zeta.$$

Hence, the approximation ratio and competitive ratio are metrics to analyze worst-case performance of offline and online algorithms, respectively. Note that these two ratios are greater than or equal to one. Hence, with $\theta \geq 0$, any ratios obtained with respect to (7) still hold with respect to (6). Therefore, in this work, we focus on analyzing the approximation and competitive ratios with respect to (7).

IV. Offline Resource Procurement and Allocation Algorithm

In this section, we propose an offline approximation algorithm when the demands in all time slots $\mathbf{d} = \{d_1, \ldots, d_T\}$ are given, which has pseudo-polynomial run time. The design of this offline algorithm will inspire the online algorithm in Section V. Furthermore, its approximation ratio provides an intermediate step to derive the competitive ratio of the online algorithm.

A. Algorithm Description

We divide the demands into $d^{\max}$ levels, where $d^{\max}$ is the peak demand, i.e., $d^{\max} := \max_t d_t$. For example, in Fig. 2, the demands are divided into 5 levels. Let d_t^l denote the demand at time t in level l, such that

$$d_t^l = \begin{cases} 1 & \text{if } d_t \geq l, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Let utilization u^l denote the number of time slots d_t is greater than or equal to l, i.e.,

$$u^l = \sum_{t=1}^{T} d_t^l. \tag{9}$$

We note that since $d_t^l \leq d_t^{l-1}$, u^l is a non-increasing function with respect to l.

The offline algorithm is described as follows. First, we consider the K non-overlapping intervals, each of τ duration, that comprise the T time period as described in Section III-D. Let $I_k, k \in \{1, \ldots, K\}$, denote the intervals. The proposed offline algorithm decides how many remote VMs should be reserved at the beginning of each interval I_k, i.e., when $t = (k-1)\tau + 1$. Let

$$u_k^l = \sum_{t \in I_k} d_t^l \tag{10}$$

denote the utilization of level l in I_k.

Consider level $l = 1$. Based on Remark 2, if a VM is reserved for the demand at this level, user requests from $l = 2$ to $w+1$ are allocated to edge VMs and the ones from $l = w+2$ to $d^{\max}$ are allocated to on-demand VMs. Otherwise, the requests from $l = 1$ to w are allocated to edge VMs and the ones from $w+1$ to $d^{\max}$ are allocated to on-demand VM. Therefore, VM reservation is justified if

$$\gamma + \lambda \sum_{j=2}^{1+w} u_k^j + p \sum_{j=w+2}^{d^{\max}} u_k^j \leq \lambda \sum_{j=l}^{w} u_k^j + p \sum_{j=w+1}^{d^{\max}} u_k^j,$$

which implies,

$$\gamma \leq \lambda u_k^1 + (p - \lambda) u_k^{w+1}.$$

More generally, consider level l when $l-1$ VMs are already reserved, the reservation at level l is justified if

$$l\gamma + \lambda \sum_{j=l+1}^{l+w} u_k^j + p \sum_{j=l+w+1}^{d^{\max}} u_k^j \leq (l-1)\gamma + \lambda \sum_{j=l}^{l+w-1} u_k^j + p \sum_{j=l+w}^{d^{\max}} u_k^j,$$

which implies

$$\gamma \leq \lambda u_k^l + (p - \lambda) u_k^{l+w}.$$

Therefore, from demand level $l = 1$ to $d^{\max}$, the edge node reserves one additional VM at each level l if and only if

$$\gamma \leq \lambda u_k^l + (p - \lambda) u_k^{l+w}. \tag{11}$$

This algorithm gives the total number of VMs that should be reserved for I_k. Then, the user requests are allocated to the three types of VMs according to (3), (4), and (5). We term this the Offline Resource Procurement and Allocation Algorithm (OfflineRPAA) and summarize it in Algorithm 1. This algorithm has $O(d^{\max}T)$ time complexity and $O(T)$ space complexity, where T is the length of time horizon and $d^{\max}$ is the peak computing demand.

B. Performance Guarantee

In this section, we show that Algorithm 1 achieves a 2-approximation ratio. First, let $\mathcal{X}' \subset \mathbb{N}^T$ denote the set of

Algorithm 1 Offline Resource Procurement and Allocation Algorithm (OfflineRPAA)

Input: Segment T into intervals $\{I_k\}_{k=1,2,\dots,K}$, each with length τ. Initiate $r_t := 0$ for all t. For each segment I_k, the edge node knows $d_t, t \in I_k$, the pricing structure's hyper-parameters γ, p, and λ.

Output: For each I_k, we compute reservation decision $r_{(k-1)\tau+1}$, allocation decisions a_t^r, a_t^o and a_t^w, $t \in I_k$.

for all intervals $\{I_k\}$ **do**
  for $l = 1$ to $l = d^{\max}$ **do**
    if $\gamma \le \lambda u_k^l + (p-\lambda) u_k^{l+w}$ **then**
      $r_{(k-1)\tau+1} \leftarrow r_{(k-1)\tau+1} + 1$.
    end if
  end for
end for
At each time slot t,

$$a_t^r = \min\{n_t, d_t\}$$
$$a_t^w = \begin{cases} (d_t - n_t)^+, & \text{if } 0 \le d_t - n_t \le w \\ w, & \text{if } d_t - n_t > w \end{cases}$$
$$a_t^o = (d_t - a_t^w - n_t)^+.$$

solutions in which reservation decisions are made only at the beginning of each interval, i.e.,

$$\mathcal{X}' = \Big\{ \{r_t\} \in \mathbb{N}^T | r_t = 0, \text{when } t \ne (k-1)\tau + 1, \; k \in \{1, \dots, K\} \Big\}. \quad (12)$$

We will show that the solution generated by Algorithm 1 achieves the smallest cost among all $\mathbf{r} \in \mathcal{X}'$ for $\mathscr{P}_1$. Then, we will show that there exists a solution $\mathbf{r}^f \in \mathcal{X}'$ that achieves a 2-approximation ratio. Hence, Algorithm 1 achieves 2-approximation ratio.

Finding a solution $\mathbf{r} \in \mathcal{X}'$ to minimize $\mathscr{P}_1$ is equivalent to the following:

$$\mathscr{P}_2 : \min_{\mathbf{r} \in \mathcal{X}'} \sum_{k=1}^{K} \Big[\lambda \sum_{t \in I_k} a_t^w + p \sum_{t \in I_k} (d_t - a_t^w - n_t)^+ + \gamma \sum_{t \in I_k} r_t \Big].$$

Since the remote reserved VMs under Algorithm 1 span only one interval, $\mathscr{P}_2$ can be decomposed into K independent sub-problems as follows.

$$\mathscr{P}_2^k : \min_{r_{(k-1)\tau+1}} \gamma r_{(k-1)\tau+1} + p \sum_{t \in I_k} (d_t - a_t^w - n_t)^+ + \lambda \sum_{t \in I_k} a_t^w. \quad (13)$$

Lemma 1. *Algorithm 1 achieves an optimal decision in each $\mathscr{P}_2^k$. In other words, Algorithm 1 provides the lowest cost for $\mathscr{P}_1$, restricted to $\mathbf{r} \in \mathcal{X}'$.*

Proof. Firstly, the cost incurred by an algorithm is equal to the sum of the cost of each level of demand. Hence, an algorithm that provides the lowest sum of all levels' cost is optimal.

Consider an interval I_k, let r_k^+ denote the optimal number of reserved VMs for I_k. From (1), we have $n_t^+ = r_k^+, \forall t \in I_k$. According to Remark 2, at t with $n_t > 0$, we should allocate requests to reserved VMs first. As a result, with $n_t^+ = r_k^+$, we should allocate d_t to r_k^+ reserved VMs first for all $t \in I_k$, i.e., all utilizations u_k^l from $l = 1$ to r_k^+ are allocated to reserved VMs without any gaps in between.

Consider level $r_k^+ + 1$ of demand, reserving one more VM at this level increases the cost within I_k. Formally, this is expressed by the following inequality.

$$(r_k^+ + 1)\gamma + \lambda \sum_{j=r_k^+ + 2}^{r_k^+ + w + 1} u_k^j + p \sum_{j=r_k^+ + w + 2}^{d^{\max}} u_k^j > r_k^+ \gamma + \lambda \sum_{j=r_k^+ + 1}^{r_k^+ + w} u_k^j + p \sum_{j=r_k^+ + w}^{d^{\max}} u_k^j,$$

which implies

$$\gamma > \lambda u_k^{r_k^+ + 1} + (p - \lambda) u_k^{r_k^+ + w + 1}.$$

Since u^l is a non-increasing function respect to l,

$$\gamma > \lambda u_k^l + (p - \lambda) u_k^{l+w}, \quad \forall l > r_k^+. \quad (14)$$

Hence, from (14) and (11), we see that Algorithm 1 does not reserve at any levels higher than level r_k^+.

Suppose Algorithm 1 reserves VMs from levels $l = 1$ to l', and $l' < r_k^+$. Consider level $l' + i, i \in \mathbb{N}$ such that $l' < l' + i \le r_k^+$. According to Algorithm 1, since a VM is not reserved at level $l' + i$, we have

$$\gamma > \lambda u_k^{l'+i} + (p - \lambda) u_k^{l'+i+w}.$$

However, the optimal solution reserves at level l, which implies

$$\gamma \le \lambda u_k^{r_k^+} + (p - \lambda) u_k^{r_k^+ + w}.$$

Thus,

$$\lambda u_k^{l'+i} + (p - \lambda) u_k^{l'+i+w} < \lambda u_k^{r_k^+} + (p - \lambda) u_k^{r_k^+ + w}. \quad (15)$$

However, since u_k^l is a non-increasing function with respect to l, (15) does not hold. As a result, Algorithm 1 reserves until reaching level r_k^+. The lemma is proven. □

Let $c^{\text{Alg 1}}$ denote the cost achieved by Algorithm 1. By applying Lemma 1 above, we obtain the following main proposition:

Proposition 1. *Algorithm 1 has 2-approximation ratio, i.e., $c^{\text{Alg 1}} \le 2c^*$.*

Proof. Let OPT denote the optimal solution where $\mathbf{r}^* = \{r_1^*, \dots, r_T^*\}$ is the optimal reservation of $\mathscr{P}_1$. There exists a solution $\mathbf{r}^f \in \mathcal{X}'$, whose reservation decisions r_t^f at $t = (k-1)\tau + 1$ are as follows:

$$r_{(k-1)\tau+1}^f = \begin{cases} \sum_{i=1}^{\tau} r_i^*, & \text{if } k = 1, \\ \sum_{i=(k-2)\tau+1}^{k\tau} r_i^*, & \text{if } k = 2, \dots, K. \end{cases} \quad (16)$$

We note that $r_{(k-1)\tau+1}^f$ is the sum of the optimal reservations in the previous interval $[(k-2)\tau+1, (k-1)\tau]$ and the current interval $[(k-1)\tau+1, k\tau]$ when $k > 1$. Then, we have

$$\sum_{t=1}^{T} r_t^f \le 2 \sum_{t=1}^{T} r_t^*. \quad (17)$$

Let n_t^* and n_t^f denote the numbers of remote reserved VMs that remain effective at time t of the optimal strategy and $\mathbf{r}^f$, respectively. Now, we compare n_t^f and n_t^*. For $t \in \{1,\dots,\tau\}$, we have

$$n_t^* = \sum_{i=1}^{t} r_i^*,$$
$$n_t^f = \sum_{i=1}^{t} r_i^f \overset{\{a\}}{=} r_1^f = \sum_{i=1}^{\tau} r_i^*,$$

where $\overset{\{a\}}{=}$ above is because $r_i^f = 0$ if $i \neq 1$. Hence, $n_t^f \geq n_t^*$, when $t \in \{1,\dots,\tau\}$. For $t = (k-1)\tau + j$, where $j \in \{1,\dots,\tau\}$ and $k > 1$, we have

$$n_{(k-1)\tau+j}^* = \sum_{i=(k-2)\tau+j+1}^{(k-1)\tau+j} r_i^*,$$
$$n_{(k-1)\tau+j}^f = \sum_{i=(k-2)\tau+j+1}^{(k-1)\tau+j} r_i^f \overset{\{b\}}{=} r_{(k-1)\tau+1}^f = \sum_{i=(k-2)\tau+1}^{k\tau} r_i^*,$$

where $\overset{\{b\}}{=}$ above is because $r_t^f = 0$ if $t \neq (k-1)\tau + 1$. For any $j \in \{1,\dots,\tau\}$, we have $(k-2)\tau + 1 \leq (k-2)\tau + j + 1$ and $k\tau \geq (k-1)\tau + j$. Hence we have

$$n_t^f \geq n_t^*, \quad \text{for} \quad t = (k-1)\tau + j, j \in \{1,\dots,\tau\}.$$

Hence,

$$n_t^f \geq n_t^*, \quad \forall t. \tag{18}$$

From (18), according to Remark 2, for both OPT and Algorithm 1, given the same demand $\mathbf{d}$, they both allocate requests to remote reserved VMs first, local VMs second, and then on-demand VM last. Therefore, we achieve

$$a_t^{w^f} \leq a_t^{w^*}, \tag{19}$$

and

$$a_t^{o^f} = (d_t - a_t^{w^f} - n_t^f)^+ \leq (d_t - a_t^{w^*} - n_t^*)^+ = a_t^{o^*}, \forall t. \tag{20}$$

Let c^* and c^f be the objective values of (7) of the optimal solution and $\mathbf{r}^f$, respectively. From (17), (19), and (20), we have

$$\begin{aligned} c^f &= \gamma \sum_{t=1}^{T} r_t^f + p \sum_{t=1}^{T} (d_t - a_t^{w^f} - n_t^{w^f})^+ + \lambda \sum_{t=1}^{T} a_t^{w^f} \\ &\leq 2\gamma \sum_{t=1}^{T} r_t^* + p \sum_{t=1}^{T} (d_t - a_t^{w^*} - n_t^*)^+ + \lambda \sum_{t=1}^{T} a_t^{w^*} \\ &\leq 2c^*. \end{aligned}$$

According to Lemma 1, since $\mathbf{r}^f \in \mathcal{X}'$, we have $c^{\text{Alg 1}} \leq c^f$. Hence, $c^{\text{Alg 1}} \leq c^f \leq 2c^*$. This proves the proposition. □

We note that although the approximation ratio that Algorithm 1 achieves is similar to that of the offline algorithm proposed in [7], the proof of Proposition 1 is substantially different and utilizes Remark 2 with the assumption of $\theta \leq \lambda' \leq p'$.

V. Online Resource Procurement and Allocation Algorithm

In this section, we consider an online strategy without any prior knowledge about the future demand. We keep track of the past demand of users and make decision at each time slot t after the current demand arrives.

A. Algorithm Description

Inspired by Algorithm 1 and Proposition 1, we again divide the time axis into intervals of length τ timeslots. Within any of such interval I_k, at each time slot $t \in I_k$, i.e., $t = (k-1)\tau + i$ for $i \in \{1,\dots,\tau\}$, we dynamically update the sequence of reservation decisions $\{r_t\}$ from the current time slot to the end of the interval. Note that since the edge node makes irrevocable reservations, at each time slot $t \in I_k$, we can only update $\{r_{t'}\}$ for $t' \in \{t,\dots,k\tau\}$. In addition, the value of r_t can only increase or remain unchanged, as we make new reservation decisions in each timeslot.

Our decision is based on the history of demand d_t, the number of previously added reserved VMs r_t, and the number of remaining active reserved VMs $n_{t'}$ for $t' \in \{t,\dots,t+\tau-1\}$. Similarly to how the offline Algorithm 1 uses (11), in the online algorithm, the edge node will reserve a VM at level l if

$$\gamma \leq \lambda \sum_{i=(k-1)\tau+1}^{t} d_i^l + (p-\lambda) \sum_{i=(k-1)\tau+1}^{t} d_i^{l+w}. \tag{21}$$

Then, the VM requests are allocated to the three types of VMs according to Remark 2. Note that (11) suggests the edge node should reserve at level l if the gain of reserving one more VM is higher than its upfront cost. In contrast, (21) suggests the edge node should reserve if the gain of the hypothetical scenario, where the edge node had reserved a VM at the beginning of interval I_k, is higher than the upfront cost.

We further note that in some intervals, if a remote reserved VM is procured at level l while there is no reserved instance at level $l-1$, the reserved instance is assigned at $l-1$. Moreover, while there is already a VM reserved at level l, the edge node will postpone to reserve at level l until the VM expires. The resultant algorithm is termed the Online Resource Procurement and Allocation Algorithm (Online RPPA) and is summarized in Algorithm 2, which is run continuously at each timeslot t.

We note in particular that, for each time t, with the knowledge of the number of remaining active reserved VM $n_{t'}$ from $t' = 1$ to $t + \tau - 1$, if (21) is satisfied, then, from lines 4 to 8 of Algorithm 2, we inspect the number of reservations from the current time t to the end of I_k, and then we reserve a VM at level l only if there exists a t' such that $n_{t'} < l$. Under this procedure, we reserve at most one VM within I_k at any given level l, to avoid redundant reservations.

Since T goes to infinity, we consider the complexity of Algorithm 2 in a single time slot. Assuming that $d^{\max}$ is given, it has $O(d^{\max}\tau)$ time complexity and $O(\tau)$ space complexity.

B. Performance Guarantee

Among the intervals of τ timeslots defined above, let $\mathcal{I}_{\text{cheap}}$ denote the set of intervals in which Algorithm 1 does

Algorithm 2 Online Resource Procurement and Allocation Algorithm (OnlineRPAA)

Input: demand d_t, corresponding I_k, previously added reserved VM $r_{t'}$ for $t' \in \{t, \dots, k\tau\}$ and the number of remaining active reserved VM $n_{t'}$ for $t' \in \{t, \dots, t+\tau-1\}$, the pricing structure's hyper-parameters γ, p, λ

Output: updated reservation decisions r'_t with $t' \in \{t, \dots, k\tau\}$, allocation decisions a_t^r, a_t^o and a_t^w

for $l = 1$ to d_t **do**
  if $\gamma \le \lambda \sum_{i=(k-1)\tau+1}^{t} d_i^l + (p-\lambda) \sum_{i=(k-1)\tau+1}^{t} d_i^{l+w}$ **then**
    for $t' = t$ to $k\tau$ **do**
      if $n_{t'} < l$ **then**
        $r_{t'} \leftarrow r_{t'} + 1$.
      end if
    end for
  end if
end for
The VM requests are assigned as follows:

$$a_t^r = \min\{n_t, d_t\}$$
$$a_t^w = \begin{cases} (d_t - n_t)^+, & \text{if } 0 \le d_t - n_t \le w \\ w, & \text{if } d_t - n_t > w \end{cases}$$
$$a_t^o = (d_t - a_t^w - n_t)^+.$$

not reserve any remote VMs, and let $\mathcal{I}_{\text{expensive}}$ denote the set of intervals in which Algorithm 1 reserves at least one remote VM. Recall that the objective value of (7) achieved by Algorithm 1 is denoted by $c^{\text{Alg 1}}$. We further let $c^{\text{Alg 2}}$ denote the part of the objective value of (7) resulting from Algorithm 2. We also let c_k denote the objective values of (7) within I_k

$$c_k = \gamma \sum_{t \in I_k} r_t + \lambda \sum_{t \in I_k} a_t^w + p \sum_{t \in I_k} (d_t - a_t^w - n_t)^+. \quad (22)$$

Since I_k are non-overlapping intervals, $c^{\text{Alg 1}} = \sum_k c_k^{\text{Alg 1}}$ and $c^{\text{Alg 2}} = \sum_k c_k^{\text{Alg 2}}$.

Lemma 2. *Within $I_k \in \mathcal{I}_{cheap}$, $c_k^{Alg\ 2} \le c_k^{Alg\ 1}$.*

Proof. Algorithm 1 does not reserve any VMs within any $I_k \in \mathcal{I}_{\text{cheap}}$, i.e., $r_t^{\text{Alg 1}} = n_t^{\text{Alg 1}} = 0, \forall t \in I_k$. From (21), Algorithm 2 also does not reserve any VM within I_k, i.e., $r_t^{\text{Alg 2}} = 0, \forall t \in I_k$, which is equivalent to

$$r_t^{\text{Alg 1}} = r_t^{\text{Alg 2}} = 0, \forall t \in I_k. \quad (23)$$

However, Algorithm 2 can reserve at any time t, not just at the beginning of each I_k. Therefore, there may be some remaining active reserved VMs from the previous interval (which necessarily is in $\mathcal{I}_{\text{expensive}}$), i.e., $n_t^{\text{Alg 2}} \ge n_t^{\text{Alg 1}}$. Hence, similar to how we obtain (19) and (20) from (18), we have, $\forall t \in I_k$,

$$\begin{aligned} a_t^{w^{\text{Alg 2}}} &\le a_t^{w^{\text{Alg 1}}}, \\ a_t^{o^{\text{Alg 2}}} &= (d_t - a_t^{w^{\text{Alg 2}}} - n_t^{\text{Alg 2}})^+ \\ &\le (d_t - a_t^{w^{\text{Alg 1}}} - n_t^{\text{Alg 1}})^+ = a_t^{o^{\text{Alg 1}}}. \end{aligned} \quad (24)$$

Moreover, by substituting (23) and (24) into (13), we have

$$\begin{aligned} &p \sum_{t \in I_k} (d_t - a_t^{w^{\text{Alg 2}}} - n_t^{\text{Alg 2}})^+ + \lambda \sum_{t \in I_k} a_t^{w^{\text{Alg 2}}} \\ \le &p \sum_{t \in I_k} (d_t - a_t^{w^{\text{Alg 1}}} - n_t^{\text{Alg 1}})^+ + \lambda \sum_{t \in I_k} a_t^{w^{\text{Alg 1}}}, \end{aligned}$$

which is equivalent to

$$c_k^{\text{Alg 2}} \le c_k^{\text{Alg 1}}. \quad (25)$$

□

Next, we consider an arbitrary $I_k \in \mathcal{I}_{\text{expensive}}$. For this case, we need to define the cost of a reservation strategy (i.e., the objective values of (7) for either Algorithm 1 or Algorithm 2) at level l within interval I_k as follows:

$$\begin{aligned} c_{k,l} = &\sum_{t \in I_k} \Big[\gamma \mathbb{I}(r_t \ge l) + d_t^l \Big(\lambda \mathbb{I}(n_t < l \le n_t + w) \\ &+ p \mathbb{I}(l > n_t + w) \Big) \Big] \quad (26) \\ = &\gamma \sum_{t \in I_k} \mathbb{I}(r_t \ge l) + \lambda \sum_{t \in I_k} d_t^l \mathbb{I}(n_t < l \le n_t + w) \\ &+ p \sum_{t \in I_k} d_t^l \mathbb{I}(l > n_t + w), \quad (27) \end{aligned}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

The following lemma indicates that this definition of $c_{k,l}$ properly separates the total cost by demand levels:

Lemma 3. *In any interval I_k, $c_k = \sum_l c_{k,l}$.*

Proof. Consider the first element of (27) and (22), we have

$$\begin{aligned} &\sum_l \gamma \sum_{t \in I_k} \mathbb{I}(r_t \ge l) \\ = &\gamma \sum_l \Big(\mathbb{I}(r_{(k-1)\tau+1} \ge l) + \dots + \mathbb{I}(r_{k\tau} \ge l) \Big) \\ = &\gamma \Big(\sum_l \mathbb{I}(r_{(k-1)\tau+1} \ge l) + \dots + \sum_l \mathbb{I}(r_{k\tau} \ge l) \Big) \\ = &\gamma \Big(r_{(k-1)\tau+1} + \dots + r_{k\tau} \Big) \\ = &\gamma \sum_{t \in I_k} r_t. \end{aligned} \quad (28)$$

Consider the second element of (27) and (22), we have

$$\begin{aligned} &\sum_l \lambda \sum_{t \in I_k} d_t^l \mathbb{I}(n_t < l \le n_t + w) \\ = &\lambda \sum_{t \in I_k} \Big[(d_t - n_t) \mathbb{I}(n_t < d_t \le n_t + w) + w \mathbb{I}(d_t > n_t + w) \Big] \\ = &\lambda \sum_{t \in I_k} a_t^w. \end{aligned} \quad (29)$$

Consider the last element of (27) and (22), we have

$$\begin{aligned} &\sum_l p \sum_{t \in I_k} d_t^l \mathbb{I}(l > n_t + w) \\ = &p \sum_{t \in I_k} (d_t - w - n_t) \mathbb{I}(d_t > n_t + w). \end{aligned} \quad (30)$$

TABLE III
POWER CONSUMPTION OF PHYSICAL PROCESSORS CORRESPONDING TO EC2 INSTANCE OFFERS.[1]

Instance type	Equivalent Physical Processor	Full Load	Hourly On-demand Price
m3.medium	Intel Xeon E5-2670	305W	$0.067
c3.large	Intel Xeon E5-2680	425W	$0.105

Moreover, when $n_t < l \leq n_t + w$, we have

$$a_t^w = d_t - n_t.$$

Hence,

$$\begin{aligned} & p\sum_{t\in I_k}(d_t - w - n_t)\mathbb{I}(d_t > n_t + w) \\ =& p\sum_{t\in I_k}(d_t - w - n_t)\mathbb{I}(d_t > n_t + w) \\ & + p\sum_{t\in I_k}(d_t - a_t^w - n_t)\mathbb{I}(n_t < d_t \leq n_t + w) \\ =& p\sum_{t\in I_k}(d_t - a_t^w - n_t)^+. \end{aligned} \tag{31}$$

From (28), (29), (30), and (31), we have $c_k = \sum_l c_{k,l}$. □

Let $c_{k,l}^{\text{Alg 1}}$ and $c_{k,l}^{\text{Alg 2}}$ be the objective values of (7) for Algorithm 1 and Algorithm 2, respectively. Then from Lemma 3, we have $c_k^{\text{Alg 1}} = \sum_l c_{k,l}^{\text{Alg 1}}$ and $c_k^{\text{Alg 2}} = \sum_l c_{k,l}^{\text{Alg 2}}$. In the next two lemmas, we compare $c_{k,l}^{\text{Alg 1}}$ and $c_{k,l}^{\text{Alg 2}}$ for two difference cases of the value of level l, which are then combined in Lemma 6 to provide a bound on the ratio between $c_k^{\text{Alg 1}}$ and $c_k^{\text{Alg 2}}$.

Lemma 4. *Within $I_k \in \mathcal{I}_{expensive}$, let l_r be the number of VMs that Algorithm 1 reserves in the first time slot of I_k. For $l \in \{1, \ldots, l_r\}$, $c_{k,l}^{Alg\ 2} \leq 3c_{k,l}^{Alg\ 1}$.*

Proof. The proof is omitted due to space limit. □

Lemma 5. *Within $I_k \in \mathcal{I}_{expensive}$, let l_r be the number of VMs that Algorithm 1 reserves in the first time slot of I_k. For $l \in \{l_r + 1, \ldots, d^{\max}\}$, $c_{k,l}^{Alg\ 2} \leq \frac{p}{\lambda} c_{k,l}^{Alg\ 1}$.*

Proof. At any levels at or above $l_r + 1$, Algorithm 1 does not reserve a remote VM. Thus, (11) is not satisfied. As a result, for any $t \in \{(k-1)\tau + 1, \ldots, k\tau\}$, (21) is also not satisfied, which implies that Algorithm 2 does not reserve any VMs. Hence, the cost resulted from Algorithm 1 and Algorithm 2 is from remote on-demand instances and local processing. Since $\lambda < p$, the cost of Algorithm 1 is lower-bounded by the local processing cost, while the cost of Algorithm 2 is upper-bounded by the cost of using remote on-demand instances. Thus, the ratio of $\frac{c_{k,l}^{\text{Alg 2}}}{c_{k,l}^{\text{Alg 1}}}$ is upper-bounded by $\frac{p}{\lambda}$. □

Lemma 6. *Within $I_k \in \mathcal{I}_{expensive}$, $c_k^{Alg\ 2} \leq \max\{3, \frac{p}{\lambda}\} c_k^{Alg\ 1}$.*

Proof. From Lemma 3, Lemma 4, and Lemma 5, for $I_k \in \mathcal{I}_{\text{expensive}}$, we have

$$\frac{c_k^{\text{Alg 2}}}{c_k^{\text{Alg 1}}} \leq \max_l \left\{ \frac{c_{k,l}^{\text{Alg 2}}}{c_{k,l}^{\text{Alg 1}}} \right\} = \max\{3, \frac{p}{\lambda}\}. \tag{32}$$

□

From Lemma 6, we obtain the following main proposition on the performance of Algorithm 2:

Proposition 2. *Algorithm 2 has $\max\{6, \frac{2p}{\lambda}\}$ competitive ratio.*

Proof. We have

$$\begin{aligned} \frac{c^{\text{Alg 2}}}{c^{\text{Alg 1}}} &= \frac{\sum_{k\in\mathcal{I}_{\text{cheap}}} c_k^{\text{Alg 2}} + \sum_{k\in\mathcal{I}_{\text{expensive}}} c_k^{\text{Alg 2}}}{\sum_{k\in\mathcal{I}_{\text{cheap}}} c_k^{\text{Alg 1}} + \sum_{k\in\mathcal{I}_{\text{expensive}}} c_k^{\text{Alg 1}}} \\ &\leq \max\left\{ \frac{c_k^{\text{Alg 2}}}{c_k^{\text{Alg 1}}} | I_k \in \mathcal{I}_{\text{cheap}}, \frac{c_k^{\text{Alg 2}}}{c_k^{\text{Alg 1}}} | I_k \in \mathcal{I}_{\text{expensive}} \right\} \\ &= \max\left\{1, \max\{3, \frac{p}{\lambda}\}\right\} \\ &= \max\{3, \frac{p}{\lambda}\}. \end{aligned}$$

From Proposition 1, Algorithm 1 has 2-approximation ratio, i.e., $c^{\text{Alg 1}} \leq 2c^*$. Hence, $c^{\text{Alg 2}} \leq \max\{6, \frac{2p}{\lambda}\}c^*$. □

Next, we consider the performance of Algorithm 2 in a special case where the local capacity at the edge node is zero. In this case, the system is reduced to the one in [7]. However, we note that Algorithm 2 is different from the online algorithm proposed in [7] because Algorithm 2 does not consider past history in the beginning of each interval. It only considers the history within each I_k. This is in contrast to [7], where at any given t, the proposed online algorithm always considers historical demand from $t - \tau + 1$ to t. Nevertheless, as shown in the following, in this case Algorithm 2 has the same competitive ratio as the online algorithm proposed in [7].

Proposition 3. *When there is no edge capacity, i.e., $w = 0$, Algorithm 2 has 4 competitive ratio.*

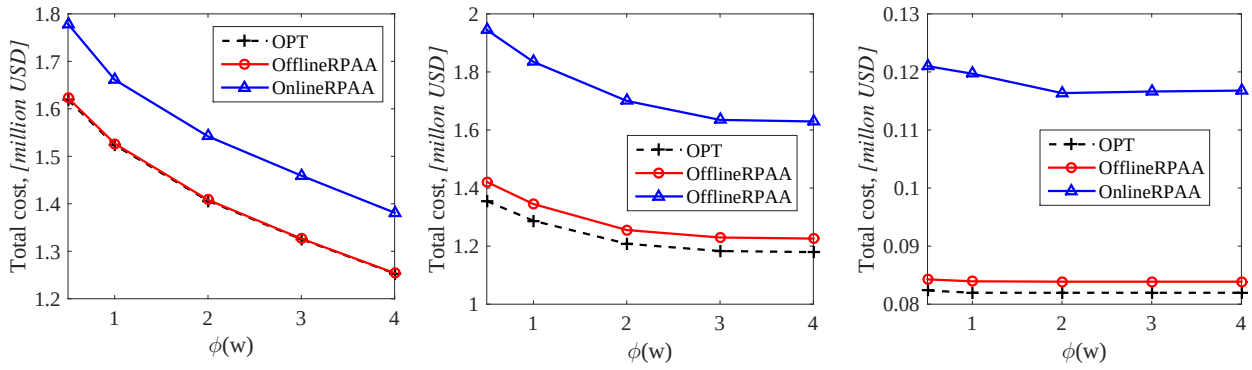
Proof. The proof is omitted due to space limit. □

VI. NUMERICAL RESULTS

Besides the approximation and competitive ratios derived in the previous sections, we numerically evaluate the performance of the proposed algorithms with extensive simulation based on the parameters specified in Amazon pricing policies [2] and Google cluster-usage traces [24]. We set $p' = \$0.067$ and $\gamma = \$1.0452$. We also consider Amazon's reserved instance "m3.medium" whose equivalent physical processor is Intel Xeon E5-2670.[1] From [30], [31], its power consumption in full utilization is 305 W, as shown in Table III. The edge processing cost λ' is set at $0.03 based on electricity usage, assuming the electricity price at $0.1 per 1 kWh.[2] Unless otherwise specified, the default value of the edge's capacity

[1]The equivalent processors are according to [29]. The power consumption of processors is measured under stress tests in [30], [31].

[2]The electricity price in US in Oct. 2017 is from ¢8.20 to ¢15.40 [32].



(a) Group: 1 (b) Group: 2 (c) Group: 3

Fig. 3. The impact of the edge node's capacities on the total cost.

is set at one standard deviation of the demand, and the default effective reservation time τ is one week. Google cluster-usage traces were measured in one month, i.e., T = one month (4 weeks). With these pricing structures, even though the edge processing is computed only from electricity cost, the competitive ratio is 6 with "m3.medium" and "c3.large". As explained in Section III, the value of θ is not important. Therefore, without loss of generality, we set $\theta = 0$.

A. Comparison Targets

We compare the performance of the two proposed algorithms with the following baselines.

1. Edge and On-Demand: Here, we process VM requests first at the edge (see Remark 2). The excess requests are offloaded to the cloud with remote on-demand VMs. This strategy is labeled as "E+Od".
2. Wang *et. al*: Here, we apply the online algorithm proposed by Wang *et. al* in [7] where the authors only consider remote reserved VMs and remote on-demand VMs. This algorithm is labeled as "Wang".
3. Edge + Wang *et. al*: Here, we process VM requests first at the edge. The excess requests are allocated by following the algorithm "Wang" above. This strategy is labeled as "E+Wang".
4. On-Demand Only: Here, all VM requests are served by remote on-demand VMs.

These algorithms are compared in different scenarios where demands have different fluctuation levels, in order to reveal the effects of the edge node's capacities and the cloud node's pricing structure on the performance of these algorithms. We also investigate the impact of the reservation period on the algorithms' performance.

B. Google Cluster-Usage Traces

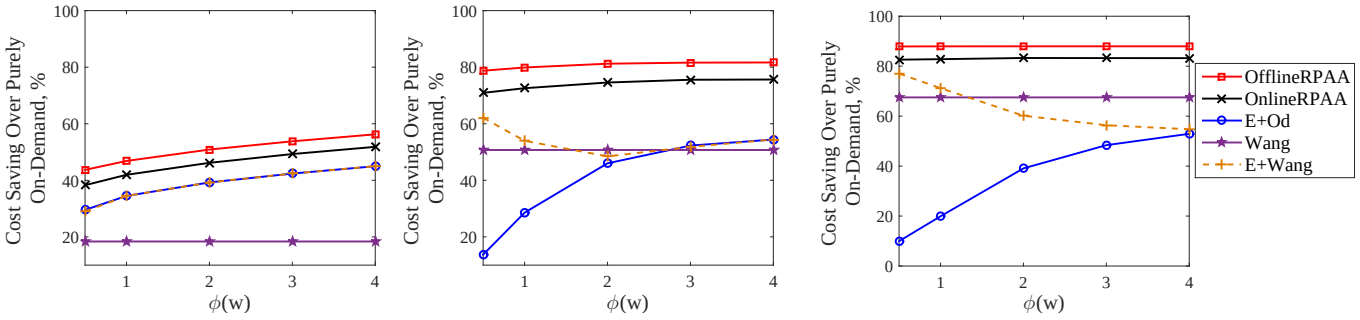
Since the workload information in public clouds is often confidential, we use Google cluster-usage traces [24] to examine the proposed algorithms in practical scenarios. We assume that Google's computing demands approximate public IaaS servers' demands [7]. Google recorded tasks arriving at one of its server clusters of about 12500 physical machines within one month in May 2011. Here, we use the revised data, version 2.1, which was updated on Nov. 11, 2014. Since user names are encrypted by strings of characters, we use function "as.factor" in the R programming language to determine the number of different strings. We find 901 users within the trace period. After that, we use function "as.numeric" in R to produce a one-to-one mapping from strings to numbers. As a result, user names are converted from strings to numbers. Tasks arrive in the time scale of μs, while the billing cycle of on-demand VMs is one hour. For simplicity, each task is assumed to be equivalent to one VM request, which takes one hour to be processed. We also assume that an instance is required to serve each VM request. Therefore, for each user, its demand curve is computed by counting the number of tasks arriving in each hour. We then analyze the users' demand within the month. Similar to [7], we also divide users into 3 groups based on their demand fluctuation level, i.e., the ratio between the user demand's standard deviation and mean.

1. Group 1 (High Fluctuation): Users in this group have demand fluctuation level greater than 5. There are 570 users in this group.
2. Group 2 (Medium Fluctuation): Users in this group have demand fluctuation levels between 1 and 5. There are 308 users in this group.
3. Group 3 (Low Fluctuation): Users in this group have demand fluctuation levels less than 1. There are 23 users in this group.

The edge node simply adds up all users' demand as the aggregate demand. Since the data is recorded in one month, the length of the aggregate demand vector is 672 timeslots in length, which is equivalent to the number of hours in 4 weeks.

C. Impact of Edge Node's Capacity

In this section, we investigate the impact of the edge node's capacity. Here, the algorithms' performance is evaluated based on their cost savings over purely allocating VM requests to remote on-demand VMs. For each group, we consider the aggregate demand of the group. Let σ denote the standard deviation of the aggregate demand, and $\phi(w) = \frac{w}{\sigma}$ be the ratio of edge node's capacity over the demand standard deviation.



(a) Group: 1 (b) Group: 2 (c) Group: 3

Fig. 4. The impact of the edge node's capacities on the cost saving percentage of algorithms over purely assigning requests to on-demand instances.

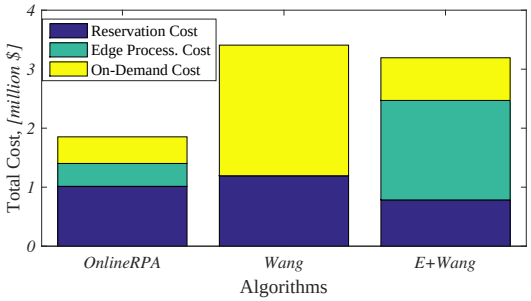


Fig. 5. The cost distribution of the three algorithms with $\tau = 1$ week at Group 2.

We consider different edge node capacities, for five different values of $\phi \in [0.5\ 1\ 2\ 3\ 4]$. We first compare the performance of the two proposed algorithms with the optimal solution found by the Branch-and-Bound algorithm. Then, we study the cost saving of the proposed algorithms over assigning on-demand VMs, in comparison with that of the baselines.

Fig. 3 shows that the proposed offline algorithm has near optimal performance, while the online algorithm also performs well. Moreover, we observe the importance of edge computing capacity. Fig. 3 suggests that the faster the demands fluctuate, the faster the total cost decays. Thus, the local VMs at the edge node serves to smooth out the fluctuation of the demands, since their usage cost is in between that of on-demand VMs and reserved VMs..

We observe in Fig. 4 that, firstly, "OfflineRPAA" and "OnlineRPAA" outperform all alternatives. Moreover, the comparison between "OnlineRPAA" and "E+Wang" suggests that an effective resource allocation algorithm should consider the edge node's parameters instead of a trivial solution such as "E+Wang". Finally, the cost savings of "OfflineRPAA" and "OnlineRPAA" over the pure on-demand strategy increase as the edge's capacity increases, especial for users in Group 1, which demonstrates the benefit of effective utilization of edge computing.

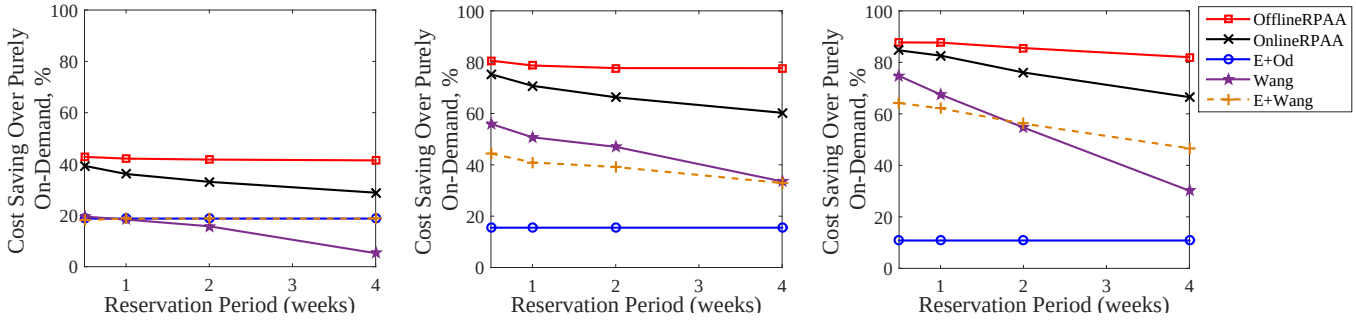
Fig. 5 explains why "OnlineRPAA" performs significantly better than "Wang" and "E+Wang", using as an example users in Group 2. As in Fig. 5, "Wang" not only reserves more than "OnlineRPAA" but also has much higher on-demand cost. It implies that many of reserved VMs in "Wang" are under utilized, while "OnlineRPAA" takes advantage of edge VMs to reduce its cost. On the other hand, in "E+Wang", VM requests are allocated to the edge VMs first and then the excess ones followed "Wang". It is observed that the cost of reservation of "E+Wang" is slightly less than that of "OnlineRPAA". However, the on-demand cost and edge processing cost of "E+Wang" are higher than those of "OnlineRPAA". Thus, we conclude that "OnlineRPAA" utilizes resources more efficiently than "E+Wang".

D. Impact of Reservation Period

In this section, we investigate the impact of the reservation period on algorithm performance. We further assume that the reservation cost γ increases proportionally with respect to the reservation period. We observe from Fig. 6 that the algorithms' performance decrease when the reservation period increases. This phenomenon is explained in Fig. 7, where we investigate the resource allocation of "OnlineRPAA" in different reservation periods, with users in Group 2 as an example. We observe that, as the reservation period increases, the number of VM requests assigned to remote reserved VMs decreases, while the number of requests allocated to the other two options increases. This implies that the number of reservations is reduced. This is reasonable since the upfront cost γ increases as τ increases. Since the edge node reserves less, the performance gain of the proposed algorithms over the pure on-demand strategy decreases. Furthermore, as shown in Fig. 6(c), the degradation of "Wang" is faster than "OnlineRPAA' and "E+Wang". This again confirms the importance of considering the edge's capacity.

VII. Conclusion

In this work, a hybrid edge-cloud system is investigated. We consider an edge node that has finite computing capacity and a cloud node that offers remote computing instances under both on-demand and reservation options. The edge node decides how to acquire computing resource from the cloud node and allocate its local and acquired resources to reduce its cost of serving users demands. An offline resource procurement and allocation solution is proposed with the prior knowledge of future demand. We then propose an online resource procurement and allocation algorithm, which makes



(a) Group: 1 (b) Group: 2 (c) Group: 3

Fig. 6. The impact of reservation periods on the cost saving percentage of algorithms over purely assigning requests to on-demand instances.

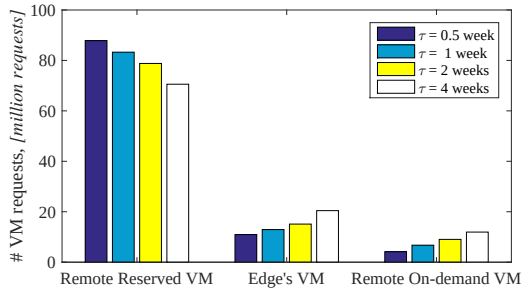


Fig. 7. VM requests allocation of OnlineRPAA with different reservation periods at Group 2.

irrevocable decision without knowledge of future demand. For both algorithms, the worst-case performance with respect to the offline optimum is provided. Numerical results show the importance of considering the edge node's computing capacity. Firstly, the existence of computing capacity at the edge can significantly reduce its cost. Secondly, we observe that under typical cloud and edge pricing structure, the proposed online algorithm, which considers the edge's cost and capacity, significantly outperforms alternative solutions, including one that always processes user requests first at the edge. Finally, when the reservation period and the upfront cost are increased proportionally, the performance of the algorithms decreases. However, the degradation of the algorithms' performance lessens if the local capacity of the edge node is increased.

REFERENCES

[1] N. C. Luong, P. Wang, D. Niyato, Y. Wen, and Z. Han, "Resource management in cloud networking using economic analysis and pricing models: A survey," *IEEE Commun. Surveys Tuts*, vol. 19, no. 2, pp. 954–1001, Second Quarter 2017.

[2] Amazon. (2018) AWS simple monthly calculator. [Online]. Available: https://calculator.s3.amazonaws.com/index.html

[3] Microsoft. (2018) Azure pricing. [Online]. Available: https://azure.microsoft.com/en-us/pricing/

[4] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. Int. Conf. High Perform. Comput., Netw. Storage Anal.,*, Nov. 2011, pp. 49:1–49:12.

[5] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in *Proc. Int. Conf. High Perform. Comput., Netw. Storage Anal.,*, Nov. 2012, pp. 22:1–22:11.

[6] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, Jan. 2013.

[7] W. Wang, D. Niu, B. Liang, and B. Li, "Dynamic cloud instance acquisition via IaaS cloud brokerage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1580–1593, Jun. 2015.

[8] X. Hu, A. Ludwig, A. Richa, and S. Schmid, "Competitive strategies for online cloud resource allocation with discounts," in *Proc. IEEE ICDCS*, Jun. 2015, pp. 93–102.

[9] Y.-J. Hong, J. Xue, and M. Thottethodi, "Dynamic server provisioning to minimize cost in an IaaS cloud," in *Proc. ACM SIGMETRICS*, Jun. 2011, pp. 147–148.

[10] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for Computation Offloading in Mobile Edge Computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[11] R. Hsu, J. Lee, T. Q. S. Quek, and J. Chen, "Reconfigurable Security: Edge-Computing-Based Framework for IoT," *IEEE Netw.*, vol. 32, no. 5, pp. 92–99, Sep. 2018.

[12] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[13] M. Satyanarayanan, "The emergence of edge computing," *Comput.*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[14] B. Liang, "Mobile edge computing," in *Key Technologies for 5G Wireless Systems*, V. W. S. Wong, R. Schober, D. W. K. Ng, and L.-C. Wang, Eds. Cambridge: Cambridge University Press, 2017.

[15] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[16] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

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

[18] Y. Lin and H. Shen, "Cloudfog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 431–445, Feb. 2017.

[19] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput*, vol. 5, no. 1, pp. 108–119, Jan. 2017.

[20] M. H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE INFOCOM*, May 2017.

[21] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 2868–2881, Dec. 2018.

[22] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin, and A. Sala, "Smoothed online resource allocation in multi-tier distributed cloud networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2556–2570, Aug. 2017.

[23] J. P. Champati and B. Liang, "One-restart algorithm for scheduling and

offloading in a hybrid cloud," in *Proc. IEEE IWQoS*, Portland, OR, USA, Jun. 2015.
[24] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.
[25] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. New York, NY, USA: Cambridge University Press, 1998.
[26] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, "Competitive randomized algorithms for non-uniform problems," *Algorithmica*, vol. 11, no. 6, pp. 542–571, Jun. 1994.
[27] R. Fleischer, "On the Bahncard problem," *Theor. Comput. Sci.*, vol. 268, no. 1, pp. 161–174, Oct. 2001.
[28] A. R. Karlin, C. Kenyon, and D. Randall, "Dynamic TCP acknowledgment and other stories about e/(e - 1)," *Algorithmica*, vol. 36, no. 3, pp. 209–224, Jul. 2003.
[29] A. Mishra, *Amazon Web Services for Mobile Developers: Building Apps with AWS*. New York, NY, USA: John Wiley & Sons, 2017.
[30] Y. Q. Chi, J. Summers, P. Hopton, K. Deakin, A. Real, N. Kapur, and H. Thompson, "Case study of a data centre using enclosed, immersed, direct liquid-cooled servers," in *Proc. SEMI-THERM*, Mar. 2014, pp. 164–173.
[31] S. Jarp, A. Lazzaro, J. Leduc, and A. Nowak, "Evaluation of the Intel Sandy Bridge-EP server processor," CERN, Geneva, Tech. Rep. CERN-IT-Note-2012-005, Mar 2012. [Online]. Available: http://cds.cern.ch/record/1434748
[32] U.S. Energy Information Administration. (2017, Oct.) Electric power monthly. [Online]. Available: https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a

Thinh Quang Dinh (S'17) received the B.Eng. degree (with first-class honours) in Electrical and Electronic Engineering (specializing in Telecommunications) from Ho Chi Minh City University of Technology (HCMUT), Vietnam in 2013, and the Ph.D. degree from Singapore University of Technology and Design (SUTD) in 2019 under the SUTD President's Graduate Fellowship. Currently, he is now a Data Scientist at Trusting Social. His main research interests are the mathematical application of optimization, machine learning and game theory to communication, networking and resource allocation problems in Mobile Edge Computing.

Ben Liang (S'94-M'01-SM'06-F'18) received honors-simultaneous B.Sc. (valedictorian) and M.Sc. degrees in Electrical Engineering from Polytechnic University in Brooklyn, New York, in 1997 and the Ph.D. degree in Electrical Engineering with a minor in Computer Science from Cornell University in Ithaca, New York, in 2001. In the 2001 - 2002 academic year, he was a visiting lecturer and postdoctoral research associate at Cornell University. He joined the Department of Electrical and Computer Engineering at the University of Toronto in 2002, where he is now a Professor. His current research interests are in networked systems and mobile communications. He has served on the editorial boards of the IEEE Transactions on Mobile Computing since 2017 and the IEEE Transactions on Communications since 2014, and he was an editor for the IEEE Transactions on Wireless Communications from 2008 to 2013 and an associate editor for Wiley Security and Communication Networks from 2007 to 2016. He regularly serves on the organizational and technical committees of a number of conferences. He is a Fellow of IEEE and a member of ACM and Tau Beta Pi.

Tony Q.S. Quek (S'98-M'08-SM'12-F'18) received the B.E. and M.E. degrees in electrical and electronics engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1998 and 2000, respectively, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2008. Currently, he is the Cheng Tsang Man Chair Professor with Singapore University of Technology and Design (SUTD). He also serves as the Acting Head of ISTD Pillar, Sector Lead of the SUTD AI Program, and the Deputy Director of the SUTD-ZJU IDEA. His current research topics include wireless communications and networking, network intelligence, internet-of-things, URLLC, and big data processing.

Dr. Quek has been actively involved in organizing and chairing sessions, and has served as a member of the Technical Program Committee as well as symposium chairs in a number of international conferences. He is currently serving as an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the Chair of IEEE VTS Technical Committee on Deep Learning for Wireless Communications as well as an elected member of the IEEE Signal Processing Society SPCOM Technical Committee. He was an Executive Editorial Committee Member for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, an Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, and an Editor for the IEEE WIRELESS COMMUNICATIONS LETTERS.

Dr. Quek was honored with the 2008 Philip Yeo Prize for Outstanding Achievement in Research, the 2012 IEEE William R. Bennett Prize, the 2015 SUTD Outstanding Education Awards – Excellence in Research, the 2016 IEEE Signal Processing Society Young Author Best Paper Award, the 2017 CTTC Early Achievement Award, the 2017 IEEE ComSoc AP Outstanding Paper Award, and the 2016-2019 Clarivate Analytics Highly Cited Researcher. He is a Distinguished Lecturer of the IEEE Communications Society and a Fellow of IEEE.

Hyundong Shin (S'01-M'04-SM'11) received the B.S. degree in electronics engineering from Kyung Hee University (KHU), Yongin-si, Korea, in 1999, and the M.S. and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 2001 and 2004, respectively. During his postdoctoral research at the Massachusetts Institute of Technology (MIT) from 2004 to 2006, he was with the Wireless Communication and Network Sciences Laboratory within the Laboratory for Information Decision Systems (LIDS).

In 2006, Dr. Shin joined the KHU, where he is now a Professor at the Department of Electronic Engineering. His research interests include quantum information science, wireless communication, and nanonetworks.

Dr. Shin was honored with the Knowledge Creation Award in the field of Computer Science from Korean Ministry of Education, Science and Technology (2010). He received the IEEE Communications Society's Guglielmo Marconi Prize Paper Award (2008) and William R. Bennett Prize Paper Award (2012). He served as a Publicity co-chair for the IEEE PIMRC (2018) and a Technical Program co-chair for the IEEE WCNC (PHY Track 2009) and the IEEE Globecom (Communication Theory Symposium 2012, Cognitive Radio and Networks Symposium 2016). He was an Editor for IEEE Transactions on Wireless Communications (2007-2012) and IEEE Communications Letters (2013-2015).