An Optimal Randomized Online Algorithm for QoS Buffer Management

LIN YANG, The Chinese University of Hong Kong, Hong Kong
WING SHING WONG, The Chinese University of Hong Kong, Hong Kong
MOHAMMAD H. HAJIESMAILI, Johns Hopkins University, USA

The QoS (Quality of Service) buffer management problem, with significant and diverse computer applications, e.g., in online cloud resource allocation problems, is a classic online admission control problem in the presence of resource constraints. In its basic setting, packets with different values according to their QoS requirements, arrive in online fashion to a switching node with limited buffer size. Then, the switch needs to make an immediate decision to either admit or reject the incoming packet based on the value of the packet and its buffer availability. The objective is to maximize the cumulative profit of the admitted packets, while respecting the buffer constraint. Even though the QoS buffer management problem was proposed more than a decade ago, no optimal online solution has been proposed in the literature. This paper contributes to this problem by proposing: 1) A fixed threshold-based online algorithm with smaller competitive ratio than the existing results; 2) an optimal deterministic online algorithm under fractional admission model in which a packet could be admitted partially; and 3) an optimal randomized online algorithm for the general problem. We consider the last result being the main contribution of this paper.

CCS Concepts: • **Theory of computation → Online algorithms**; • **Networks →** *Bridges and switches*;

Additional Key Words and Phrases: Online algorithm; QoS buffer management; randomization

ACM Reference Format:
Lin Yang, Wing Shing Wong, and Mohammad H. Hajiesmaili. 2017. An Optimal Randomized Online Algorithm for QoS Buffer Management. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 2, Article 36 (December 2017), 26 pages. https://doi.org/10.1145/3152545

1 INTRODUCTION

QoS buffer management with multiple packet values is a classic online problem in computer networks, with practical significance [1, 6, 20]. The classic example is the DiffServ (differentiated service) networks, in which packets with different QoS requirements are associated with different quantized values, which characterize the profit earned by switches if the packets are successfully delivered. When the network is congested, the switches are not able to admit all arriving packets due to limited buffer capacity. Hence, to maximize its profit, the switch must decide to admit packets with higher values. Despite its classic application, the QoS buffer management problem could be considered as a general admission control problem in several state-of-the-art applications. As an example, we stress the value-based cloud resource allocation with limited computation

Authors' addresses: Lin Yang, The Chinese University of Hong Kong, Shatin, NT, Hong Kong, yl015@ie.cuhk.edu.hk; Wing Shing Wong, The Chinese University of Hong Kong, Shatin, NT, Hong Kong, wswong@ie.cuhk.edu.hk; Mohammad H. Hajiesmaili, Johns Hopkins University, 3400 N Charles St. Baltimore, MD, USA, hajiesmaili@jhu.edu.

2476-1249/2017/12-ART36 $15.00
https://doi.org/10.1145/3152545

capacities [3, 30], in which the online jobs with different valuations must be either admitted or rejected upon their arrival based on their values and the utilization of the cloud servers.

In offline scenario in which the arrival and departure of packets are known in advance, the QoS buffer management is simple and can be solved using a linear program. However, in a real network environment many unpredictable factors impact the profile of arriving packets, hence, offline algorithms are not practical. Thus, the main research effort has been focused on online settings in which the arrival and departure profile of packets are not known in advance. The existing algorithms usually follow the well-known competitive analysis framework [10] that tries to achieve a bounded performance as compared to the offline optimum without relying on future information.

The first study on QoS buffer management with multiple values appeared as early as in 2000 [1, 20]. The work was motivated by rising trends of differentiated services in networks. The early study was rapidly augmented by proposals of various problem settings [4, 7, 11, 14, 16, 17, 21, 22, 24, 26]. These problem settings can be roughly categorized according to the operation rules on the buffer or queue into the FIFO preemptive model and the non-preemptive model. In the FIFO preemptive model, packets which have been buffered in the queue are served in a FIFO manner, and can be discarded. In the non-preemptive model, the admitted packets cannot be ejected. There are also many interesting works investigating extensions of the FIFO preemptive or non-preemptive model. For example, in [11], the authors further take into account the heterogeneous packet processing time for different traffic. In [14, 16, 17], the authors address a model which involves a departure deadline for each transferred packet. In [7, 22, 26], the basic model is extended to the multiple input queues. In Section 2, the details of existing results in the literature is discussed.

1.1 Our Results and Adopted Techniques

This paper focuses on the non-preemptive QoS buffer management problem in online setting. All the proposed algorithms have a common structure which set a threshold value to either admit or reject the packets. We refer to these algorithms as *threshold-based* online algorithms. In the followings, we summarize the main results and the conceptual framework of our solution design.

▷ In Section 4, we study a simplified problem that assumes no packets departure, i.e., the admitted packets stay at the buffer permanently. For the simplified problem which theoretically is quite similar to online knapsack problem [8, 9], we design an algorithm to adopt a fixed threshold value for every state of the queue length of the buffer. An arriving packet is admitted only when its value is above or equal to the corresponding threshold. By optimizing the threshold values, we ultimately achieve a closed-form competitive ratio, which is proved to be the optimal among all deterministic online algorithms. While using several tailored techniques suitable for the rest of the paper, our results match to the state-of-the-art result for online knapsack problem [8]. Subsequently, we extend the threshold-based strategy of the simplified problem to the original problem, with packet departure. By optimizing the threshold values, we achieve an online algorithm that attains a better competitive ratio than the existing works [1, 6, 20] and hence is of interest by itself.

▷ In Section 5, our goal is to propose a randomized online algorithm with the optimal competitive ratio. Toward this, in Section 5.1, we first relax the assumption of the discrete packet admission, i.e., either admit or reject the packet. In the relaxed model, the switching node can admit a fraction of the packet and in turn receive the reward proportionally. Then, we propose a novel strategy that builds a set of virtual sub-queues with unit capacity for each departed packet. By defining virtual sub-queues, we can track the history of packet departure, facilitating to potentially improve the competitiveness of the online algorithm.

By leveraging the idea of sub-queue construction, we then propose an algorithm that maintains a state vector consisting of the queue length of each sub-queue and associates each virtual sub-queue with a threshold-based admission strategy, a scaled version of the one proposed in Section 4. Then, the admitted portion of the arriving packet will be properly allocated among sub-queues in a *water-filling* manner. The analysis shows that the proposed algorithm achieves a competitive ratio of $\left\lceil 1 + (\ln \theta + 1)\frac{\theta}{D} \right\rceil \cdot (\ln \theta + 1)$, where θ is the packet value fluctuation ratio and D is a parameter that determines the granularity of the fractional model. With sufficiently large D, the competitive ratio approximates the known lower bound of $\ln \theta + 1$ for deterministic online algorithms [5]. Different from the *Selective Barrier* policy in [5] (with competitive ratio of $O(\ln^2 \theta) + \ln \theta + 2$ for small θ) and [6] (with competitive ratio of $e \times \lceil \ln \theta \rceil$), which set a fixed threshold value only based on the state of queue length, the proposed algorithm adopts an independent threshold-based admission strategy for each virtual sub-queue, which corresponds to one unit of buffering budget.

▷ In Section 5.2, we propose a randomized rounding approach to extend the result of the fractional admission model to the original discrete model. First, we prove that the optimal competitive ratio of any randomized online algorithm for the discrete case is lower bounded by $\ln \theta + 1$, which is also the tight lower bound of the online algorithms for the fractional admission model as well. This result shows that the randomization will not "outperform" the optimal online algorithm for the fractional admission model. Based on this observation, it is natural to seek a randomized scheme to keep the expected queue length equal to that of the fractional case. Our proposed randomized algorithm achieves the same competitive ratio as the optimal online algorithm for the fractional case, meeting the optimal lower bound, $\ln \theta + 1$, thereby it is the optimal online algorithm. The main endeavor is to properly design the admission probabilities based on the actions taken by the optimal online algorithm for the fractional admission model.

The rest of the paper is organized as follows. Section 2 reviews the literature. In Section 3, we introduce details on the problem formulation as well as the notation used in this paper. In Sections 4 and 5, we introduce the deterministic and randomized online algorithms respectively. Concluding remarks are provided in Section 6.

2 RELATED WORK

2.1 FIFO Preemptive Model

The paper [20] deals with a preemptive single-queue model where the admitted packets can be discarded. The authors study a class of greedy algorithms which discard packets with the lowest value when an overflow occurs. Then, competitive ratio of the greedy algorithm is analyzed. Following [20], many other papers aimed to find better algorithms with lower competitive ratios. Generally speaking, the state-of-the-art result on the competitive ratio for a preemptive model is 1.732 [15], yet no optimal solution has been proposed. For a special case with only two different packet values, [15] introduces a deterministic strategy and proves that this strategy achieves an optimal competitive ratio of 1.282. In addition to the single-queue model, [7] and [22] study QoS buffer management within multiple queues, achieving a competitive ratio smaller than 2 for a special case where only two packet values are involved. For other work on preemptive buffer management, readers can refer to [4, 21, 24, 25], as well as the survey paper [19].

2.2 Non-Preemptive Model

For non-preemptive buffer management, The authors in [1] provided the first study of a two-value model. In their problem setting, packets are tagged as either being a high priority packet or a low priority packet. Specifically, they assign a benefit of $\alpha \geq 1$ to every high priority packet and a benefit of 1 to every low priority packet. Then, a general lower bound of $(2\alpha - 1)/\alpha$ for the two-value setting is proved. Then, [6] proposes an algorithm that can achieve the above lower bound for the competitive ratio. The two-value setting can be characterized as a special case of a general buffer management problem where packets are allowed to take arbitrary value in a particular region

$[m, M]$. This generalization makes the algorithm design far more challenging. In [5], the authors prove that the optimal competitive ratio for deterministic online algorithms is lower bounded by $\ln\theta + 1$, where $\theta = M/m$ is the ratio between the maximum and minimum packet value. In [32], the author provides a lower bound of the competitive ratio for any online algorithm (deterministic or randomized), which is $\frac{1}{2}\ln\theta + 1$. [6] presents two online policies, the *Round-Robin* and *Selective Barrier* policy which set a linear non-decreasing threshold function with respect to the queue length, showing that they are both $e\lceil\ln\theta\rceil$-competitive. [5] improves the above results for small θ by proposing the *smooth selective barrier* policy. When θ is small (specifically, $\theta < e$), the competitive ratio of *smooth selective barrier* is proved to be $\ln\theta + 2 + O(\frac{\ln^2\theta}{B})$, where B is the buffer capacity. In spite of the above works, no optimal online algorithms, either deterministic or randomized have been proposed since this problem was first formulated. In this paper, one of our most important contributions is to introduce a randomized online algorithm for the non-preemptive model which can be proved to be optimal.

2.3 Related Theoretical Problems and Timely Applications in a Broader Background

The investigated problem model in this paper has intrinsic relations to many classic computer science problems, such as the *time series search problem* [23], the *one-way trading problem* [12, 13], the *multiple-choice secretary problem* [2, 18] and the *online knapsack problem* [8, 9]. The substantial difference is that in addition to the uncertainty in packet arrival (which is the same in above problems), the QoS buffer management problem comes with another uncertainty in the network resource supply (that corresponds to the packet departure). This makes the adversary have more flexibility to make worst input by using two sources of uncertainty.

Almost all aforementioned classic problems have timely applications in the recent active research topics. For example, [30] is an example of a natural extension of online knapsack problem in the cloud resource allocation problem. The pricing strategy presented in [30] is close to our profit-based admission control with a social welfare maximization goal. Similarly, [28] represents the first online combinatorial auction designed for the cloud computing paradigm. Both [30] and [28] are the natural extensions of the online knapsack problem. Both works, however, fail to incorporate the latter uncertainty in provisioning of the network resource. In another recent study [29], an online offering strategy is proposed in an hour-ahead electricity market with intermittent renewable supply. This problem is similar to our problem in the sense that both market price and renewable supply arrive online. The problem in [29] can be considered as a fractional version of the QoS buffer management problem (see Section 5.1). Different from [29], the solution design in this paper proves the optimality of the proposed online algorithm.

3 PROBLEM FORMULATION AND PRELIMINARIES

3.1 Problem Formulation

We partition the time horizon into slots according to the arrival time of the packets. Specifically, a time slot begins just before a new packet arrives to the switching node, and ends before the next packet arrival. By this definition, each time slot contains exactly one packet arrival.

Suppose that the size of packets are identical and the buffer can store at most B packets. The number of packet departures at time slot t is denoted by $u(t)$. By $v(t) \in [m, M]$, we denote the value of the arriving packet at the beginning of t-th time slot, and M and m denote the maximum and minimum packet values, respectively. Let $\theta = M/m$ be the value fluctuation ratio, which plays a critical role in the competitive analysis. Both $v(t)$ and $u(t)$ are exogenous inputs controlled under an adversary strategy. The online algorithm must decide to either admit or reject a packet upon its arrival. We represent the binary decision variable by $x(t) \in \{0, 1\}$, where $x(t) = 1$ represents admission of the packet; 0 otherwise. Finally, $b(t)$ denotes the buffer level, i.e., the number of packets in the buffer, at the end of time slot t, and is expressed by

$$b(t) = [b(t-1) + x(t) - u(t)]^+ ,$$

where $[\cdot]^+$ denotes projection onto the nonnegative orthant. The number of packets in the buffer must satisfy the buffer capacity constraint, i.e.,

$$b(t-1) + x(t) \leq B.$$

The objective is to maximize the profit of the switching node, i.e., the sum of value of admitted packets, over the time horizon, $\mathcal{T} = [1, T]$. Mathematically, the problem is formulated as follows:

$$\begin{aligned} \max \quad & \sum_{t\in\mathcal{T}} v(t)x(t) \\ \text{s.t.} \quad & b(t) = [b(t-1) + x(t) - u(t)]^+ , \quad \forall t \in \mathcal{T}, \\ & b(t-1) + x(t) \leq B, \quad \forall t \in \mathcal{T}, \\ \text{var.} \quad & x(t) \in \{0,1\}. \end{aligned} \tag{1}$$

In our analysis, we assume that the initial state of the buffer is 0, i.e., $b(0) = 0$. In the online context, the exogenous inputs $v(t)$ and $u(t)$, and the ending time T are not known in advance, and we do not rely on any stochastic modeling of the exogenous inputs.

Note that Problem (1) can be considered as a natural extension the *online knapsack* problem[1] [8, 9, 31], and the category of conversion problems in financial markets [27]; some well-known variants are the *time series search* problem [23], the *one-way trading* problem [12, 13] and the *secretary problem* [2, 18]. [2] The exogenous input $v(t)$ in Problem (1) is similar to the item values in the online knapsack problem and sequential online price in the conversion problems. However, $u(t)$ is another exogenous input to Problem (1) which does not exist in the aforementioned problems. In terms of competitive design, existence of two set of exogenous inputs, i.e., $v(t)$ and $u(t)$, empowers the adversary to construct worst-case instances in a larger space, potentially resulting in a worse competitive ratio. Hence, online algorithm design becomes more challenging, since the adversary is more powerful. We refer to Section 2.3 for detailed discussions regarding similar problems.

3.2 Competitive Ratio

An instance $\boldsymbol{\omega} \in \Omega$ refers to an input instance including the value $v(t)$ of arriving packets and departure rates $u(t)$ over $[1, T]$, i.e.,

$$\boldsymbol{\omega} \overset{\text{def}}{=} [\omega(t) = (v(t), u(t))]_{t\in\mathcal{T}},$$

and Ω is the set of all possible instances, i.e.,

$$\Omega \overset{\text{def}}{=} \left\{ [(v(t), u(t))]_{t=1:T} : v(t) \in [m, M], u(t) \geq 0, T \in \mathbb{Z}^+ \right\}.$$

The performance measure of online algorithm is *competitive ratio* [10], which refers to the maximum ratio of the profit earned by the OPTimal offline solution (OPT) and a particular online algorithm $\mathcal{A}$ under any input instances, i.e.,

$$\text{CR}(\mathcal{A}) \overset{\text{def}}{=} \max_{\boldsymbol{\omega}\in\Omega} \frac{\text{Prof}_{\text{OPT}}(\boldsymbol{\omega})}{\text{Prof}_{\mathcal{A}}(\boldsymbol{\omega})},$$

[1] In the online knapsack problem, the items with different weights and values, i.e., $(w(t), v(t))$ arrive online and a feasible solution is any subset $\mathcal{S}$ of items such that $\sum_{t\in\mathcal{S}} w(t) < B$, where B is the knapsack capacity. The goal is to maximize the value of selected items, i.e., $\sum_{t\in\mathcal{S}} v(t)$.

[2] In the one-way trading problem, a trader needs to exchange from one currency to another currency, given a time-varying exchange rates arriving online. The trader can decide to accept the current price or wait for the more attractive prices in future. The time series search problem and secretary problem are also quite similar [13, 23].

where $\mathrm{Prof}_{\mathrm{OPT}}(\boldsymbol{\omega})$ and $\mathrm{Prof}_{\mathcal{A}}(\boldsymbol{\omega})$ are respectively the profit obtained by the optimal offline solution and the online algorithm $\mathcal{A}$ when the input instance is $\boldsymbol{\omega}$. For a randomized online algorithm $\mathcal{R}$, we assume the adversary is *oblivious* and define the *expected competitive ratio* as the following:

$$\mathrm{ECR}(\mathcal{R}) \overset{\mathrm{def}}{=} \max_{\boldsymbol{\omega}\in\Omega} \frac{\mathrm{Prof}_{\mathrm{OPT}}(\boldsymbol{\omega})}{E[\mathrm{Prof}_{\mathcal{R}}(\boldsymbol{\omega})]}.$$

3.3 Definitions and Notations Related to Competitive Analysis

By $b_{\boldsymbol{\omega}}^{\mathcal{A}}(t)$, we denote the queue length at slot t under $\mathcal{A}$ and a particular instance $\boldsymbol{\omega}$. Let b denote the maximum queue length that $\mathcal{A}$ reaches under $\boldsymbol{\omega}$ over the time horizon, i.e., $\max_{t\in\mathcal{T}} b_{\boldsymbol{\omega}}^{\mathcal{A}}(t) = b$. By this definition, we can partition the universal set of input instances, denoted by Ω, into multiple separate subsets as follows:

$$\Omega = \bigcup_{b\in\{1,2,\ldots,B\}} \Omega_b^{\mathcal{A}},$$

where

$$\Omega_b^{\mathcal{A}} \overset{\mathrm{def}}{=} \left\{\boldsymbol{\omega}\in\Omega : \max_{t\in\mathcal{T}} b_{\boldsymbol{\omega}}^{\mathcal{A}}(t) = b\right\},$$

represents the set of all input instances that result in the maximum queue length b by executing algorithm $\mathcal{A}$.

Definition 1. *Define the local competitive ratio* $\mathrm{CR}_b(\mathcal{A})$ *under the subset of input instances* $\Omega_b^{\mathcal{A}}$ *as*

$$\mathrm{CR}_b(\mathcal{A}) \overset{\mathrm{def}}{=} \max_{\boldsymbol{\omega}\in\Omega_b^{\mathcal{A}}} \frac{\mathrm{Prof}_{\mathrm{OPT}}(\boldsymbol{\omega})}{\mathrm{Prof}_{\mathcal{A}}(\boldsymbol{\omega})}.$$

Given Definition 1, we redefine $\mathrm{CR}(\mathcal{A})$ as

$$\mathrm{CR}(\mathcal{A}) = \max_{b\in\{1,2,\ldots,B\}} \mathrm{CR}_b(\mathcal{A}).$$

We also use the following expressions for an input instance $\boldsymbol{\omega} = [(v_1,u_1),(v_2,u_2),\ldots,(v_T,u_T)]$:

(1) The notation $\times n$ is used to represent repeated input segments. Specifically, $(v,u)\times n$ (or $\boldsymbol{\omega}\times n$) signifies the input tuple (v,u) (or input segment $\boldsymbol{\omega}$) will repeatedly appear n times in the subsequent time slots.
(2) The concatenation of $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ is denoted by $\boldsymbol{\omega}_1+\boldsymbol{\omega}_2$ and expressed by

$$\boldsymbol{\omega}_1+\boldsymbol{\omega}_2 = [\omega_1(1),\omega_1(2),\ldots,\omega_1(T_1),\omega_2(1),\omega_2(2),\ldots,\omega_2(T_2)].$$

4 A SIMPLIFIED ONLINE PROBLEM AND THE THRESHOLD-BASED ALGORITHM

The QoS buffer management problem involves two exogenous inputs $v(t)$, $u(t)$. In addition, the ending time slot T is another exogenous parameter controlled by the adversary. To analyze Problem (1), we first investigate a simplified version of the original problem by setting the exogenous input $u(t)$ to be zero in the entire time horizon. In other words, we assume that there is no packet departure during the time horizon, i.e., $u(t) = 0$, for $t = 1,2,\ldots,T$.

In the simplified setting without packet departure, the problem is similar to the online knapsack problem or k-max search problem except for some detailed settings. To design an online solution for the simplified scenario, we follow a well-established design approach for these problems and explore a class of deterministic online algorithms which are *threshold-based.* The main idea of the threshold-based strategies is that for any state of the buffer, there is a fixed threshold and the incoming packet will be admitted if its value is greater than or equal to the corresponding threshold. The analysis is then focused on demonstrating that by optimizing the threshold values, the threshold-based strategy can achieve the optimal competitive ratio among all deterministic online algorithms for the simplified problem.

4.1 Threshold-Based Policy Design for a Simplified Problem

By simplifying Problem (1) and setting $u(t) = 0, t \in \mathcal{T}$, we formulate the following problem:

$$\begin{aligned} \max \quad & \sum_{t=1}^{T} v(t)x(t) \\ \text{s.t.} \quad & \sum_{t=1}^{T} x(t) \le B. \\ \text{var.} \quad & x(t) \in \{0,1\}. \end{aligned} \tag{2}$$

In Problem (2), the exogenous inputs controlled by the adversary strategy are the packet value $v(t)$ and the ending time of the investigated time period T. Correspondingly, an input instance for Problem (2) can be simplified as $[v(1), v(2), \ldots, v(T)]$, since $u(t) = 0$ for all $t \in \mathcal{T}$. The above formulation can be roughly explained in the one-way trading or k-max search setting [12, 13], that the online player is required to convert one asset into another, e.g., dollars for yen, based on current conversion rate. In the one-way trading $v(t)$ corresponds to the conversion rate at time slot t and $x(t)$ is the binary action of the player (buy or not). The only difference between Problem (2) and those two conversion problems lies in the ending time of the game. In the one-way trading problem, the game ends when all the units of assets are sold; and in the k-max search setting, the player is required to complete the transaction in a given time interval. While the formulation of Problem (2) implies that the ending time is determined by the adversary.

Another similar problem to Problem (2) is the *online knapsack problem* [8, 9, 31]. Problem (2) could be considered as the online knapsack problem with identical items (packets) in size, but, with different values. It is worth mentioning that our result in the rest of this section for problem (2) matches the state-of-the-art result for the online knapsack problem [31] where an optimal threshold-based policy is proposed. The optimality of the proposed solution in [31], however, is based on the assumption that the size of each item is much smaller than the knapsack capacity. In this section, we relax this assumption and design a threshold-based online algorithm that achieves the optimal competitive ratio and analyze it using a different analysis technique than the one used in [31]. More specifically, the analysis in [31] basically shows that for a given threshold function, the optimal competitive ratio is achieved. Our approach, in contrast, establishes a "forward policy design" principle that can derive the optimal thresholding policy by leveraging the concept of local competitive ratios as defined in Definition 1. As a result, our work facilitates the analysis of other extended problems (see Section 4.2 for the extension to the original problem).

For Problem (2), we device a *threshold-based* online algorithm called OnAlg, which is defined by means of a series of non-decreasing threshold values v_i, $i = 1, 2, \ldots, B$. For convenience, we categorize those v_is of the same value into a single *step*. The goal is to design the optimal values of v_i as a function of queue length to specify the minimum value to admit the i-th packet.

Recall that $\Omega_b^{\mathsf{OnAlg}}$ is the subset of input instances that result in the maximum queue length b upon executing the online algorithm OnAlg. The following lemma characterizes a critical property for the worst instance in $\Omega_b^{\mathsf{OnAlg}}$.

Lemma 4.1. *Assume* $\textsc{cr}_b(\mathsf{OnAlg}) > 1$ *and* $\boldsymbol{\omega} = [v(t)]_{t\in\mathcal{T}}$ *is a worst instance in* $\Omega_b^{\mathsf{OnAlg}}$. *Then at any slot* t *when* OnAlg *and* OPT *buffer packet simultaneously,* $v(t)$ *is exactly equal to the threshold of* OnAlg.

Proof. We prove Lemma 4.1 by contradiction.

Assume that a worst instance is $[v(t)]_{t\in\mathcal{T}}$. Then suppose there exists a time slot t such that OnAlg and OPT buffer packet simultaneously and $v(t)$ is larger than the threshold value of OnAlg, i.e., $v(t) > v_i$, $i = b(t-1)+1$. Now we present the following input instance to OnAlg:

$$[v(1),\dots,v(t-1)] + [v_i] + [v(t+1),\dots,v(T)].$$

Under the new instance, the total number of buffered packets and buffering time slots of OnAlg keep unchanged. The profit earned by OnAlg decreases by $v(t) - v_i$, while profit decrement of OPT is less than or equal to $v(t) - v_i$. The local competitive ratio $\text{CR}_b(\text{OnAlg})$ is larger than 1, so the above instance results in a larger competitive ratio. This contradicts the assumption that $\boldsymbol{\omega}$ is the worst instance in Ω_b^{OnAlg}. We complete the proof. □

The next lemma characterizes upper bounds for local competitive ratios of OnAlg.

Lemma 4.2. *Assume the threshold-based algorithm* OnAlg *is defined by non-decreasing threshold values v_i that satisfies the condition $v_1 = m$. If the length of the first step is l, then*

(1) *for the subset Ω_b^{OnAlg} where $b < l$, the local competitive ratio is 1, and*
(2) *for $b \ge l$, the local competitive ratio within the subset Ω_b^{OnAlg} satisfies*

$$\text{CR}_b(\text{OnAlg}) \le \frac{v_{b+1}B}{\sum_{i=1}^{b} v_i}.^{3} \tag{3}$$

Proof. For Lemma 4.2, we have the following analysis:

(1) For $b < l$, the threshold value of OnAlg is always equal to m over $\mathcal{T}$. That means OnAlg buffers all packets and obtain the same profit as OPT. In this case, the local competitive ratio is 1.
(2) Suppose $b \ge l$. If $\text{CR}_b(\text{OnAlg}) = 1$, the case is trivial and Equation (3) definitely holds. We only consider the case that $\text{CR}_b(\text{OnAlg}) > 1$. For an instance within Ω_b^{OnAlg}, the threshold values of OnAlg are always less than or equal to v_{b+1} ($v_{b+1} > m$), since the maximum queue length is b and v_i are non-decreasing. Assume $\boldsymbol{\omega}$ is a worst instance lying in Ω_b^{OnAlg}. Then, according to Lemma 4.1, a packet value under $\boldsymbol{\omega}$ will be exactly equal to the threshold value of OnAlg when OPT and OnAlg buffer this packet simultaneously. Moreover, when only OPT buffers a packet, it is obvious that the packet value is less than the threshold value of OnAlg. That means, under $\boldsymbol{\omega}$, all the packets buffered by OPT are of values less than or equal to v_{b+1}. Thus, the profit earned by OPT under $\boldsymbol{\omega}$ is at most $v_{b+1}B$. Meanwhile, the minimum profit earned by OnAlg is at least $\sum_{i=1}^{b} v_i$ due to its threshold-based admission strategy. Thus, we have proved that the largest profit ratio within subset Ω_b^{OnAlg} is at most $v_{b+1}B/\sum_{i=1}^{b} v_i$.

This completes the proof. □

For $b > l$, consider the following instance with increasing packet values:

$$[v_1, v_2, \dots, v_b, (v_{b+1} - \delta) \times B],$$

[3] For consistence, we define $v_{B+1} = M$.

under which the profit earned by OPT is $(v_{b+1} - \delta)B$ and the profit earned by OnAlg is $\sum_{i=1}^{b} v_i$. The worst-case profit ratio shown in Equation (3) can be realized by the above instance with $\delta \to 0$, i.e.,

$$\textsc{cr}_b(\mathsf{OnAlg}) \geq \lim_{\delta \to 0} \frac{(v_{b+1} - \delta)B}{\sum\limits_{i=1}^{b} v_i}.$$

Combining with Lemma 4.2, we get

$$\textsc{cr}_b(\mathsf{OnAlg}) = \frac{v_{b+1}B}{\sum\limits_{i=1}^{b} v_i}, \text{ for } b \geq l. \tag{4}$$

According to Lemma 4.2 and Equation (4), the worst case occurs among subsets $\Omega_b^{\mathsf{OnAlg}}$, $b \geq l$, so the competitive ratio of OnAlg is

$$\textsc{cr}(\mathsf{OnAlg}) = \max_{b=l,l+1,\dots,B} \frac{v_{b+1}B}{\sum\limits_{i=1}^{b} v_i}.$$

Conditioning on the length of the first step l, the minimum competitive ratio, which is denoted by $\textsc{cr}(\mathsf{OnAlg}|l)$ can be obtained by optimizing the threshold values $v_{l+1}, v_{l+2}, \dots, v_B$:

$$\begin{array}{lll} \min & y & \\ \text{s.t.} & y \geq (v_{b+1}B) / \sum\limits_{i=1}^{b} v_i, & b = l, l+1, \dots, B. \\ \text{vars.} & y,\ m \leq v_b \leq M, & b = l+1, l+2, \dots, B. \end{array} \tag{5}$$

The following lemma gives a necessary condition for $\textsc{cr}(\mathsf{OnAlg}|l)$ to achieve its minimum value.

Lemma 4.3. *$\textsc{cr}(\mathsf{OnAlg}|l)$ achieves its minimum value only if the following expression holds:*

$$\frac{v_{l+1}B}{ml} = \frac{v_{l+2}B}{ml + v_{l+1}} = \cdots = \frac{MB}{ml + \sum\limits_{i=l+1}^{B} v_i}. \tag{6}$$

In the next step, we show that the minimum value of problem (5) is at least $\ln\theta + 1$, which provides a lower bound for the competitive ratio of OnAlg. By Equation (6), we can represent B as

$$B = \left(\frac{v_{l+1}}{m} + \frac{v_{l+2} - v_{l+1}}{v_{l+1}} + \frac{v_{l+3} - v_{l+2}}{v_{l+2}} + \cdots + \frac{M - v_B}{v_B}\right)\frac{m}{v_{l+1}}l.$$

Thus, the competitive ratio of OnAlg with the first step length being l can be expressed as

$$\textsc{cr}(\mathsf{OnAlg}|l) = 1 + \frac{v_{l+1} - m}{m} + \frac{v_{l+2} - v_{l+1}}{v_{l+1}} + \cdots + \frac{M - v_B}{v_B}.$$

Let $S = \frac{v_{l+1}-m}{m} + \frac{v_{l+2}-v_{l+1}}{v_{l+1}} + \frac{v_{l+3}-v_{l+2}}{v_{l+2}} + \cdots + \frac{M-v_B}{v_B}$. Since the value of S is equal to the size of the shaded area in Figure 1, we have $\textsc{cr}(\mathsf{OnAlg}|l) > 1 + \ln\frac{M}{m} = 1 + \ln\theta$.

According to the results in Lemma 4.3, the following theorem follows.

Theorem 4.4. *Given a non-decreasing threshold values with the length of the first step being l, the optimal competitive ratio that can be achieved by* OnAlg, *denoted by $r^{(l)}$ satisfies*

$$\left(\frac{r^{(l)} + B}{B}\right)^{B-l+1} - \left(\frac{r^{(l)} + B}{B}\right)^{B-l} - \frac{\theta}{l} = 0. \tag{7}$$

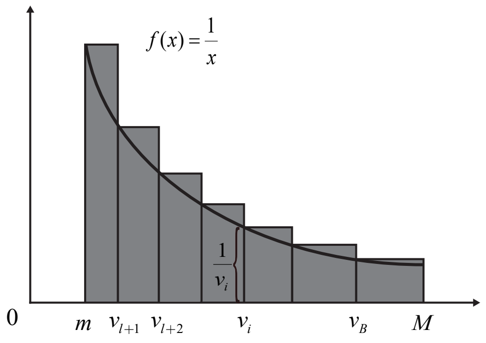

Fig. 1. Visualized expression of S.

Assuming l^* is the optimal solution to minimize the above function and r^* is the corresponding optimal value, the optimal threshold values are

$$v_{l^*+1} = \frac{r^* m l^*}{B},$$
$$v_{l^*+i+1} = v_{l^*+1}\left(\frac{M}{v_{l^*+1}}\right)^{\frac{i}{B-l^*}}, \quad \text{for } i = 1, 2, \ldots, B - l^*.$$

Proof. According to Lemma 4.3, we have

$$v_{l+1} = \frac{r^{(l)} m l}{B}.$$

Moreover,

$$\frac{v_{l+2} - v_{l+1}}{v_{l+1}} = \frac{v_{l+3} - v_{l+2}}{v_{l+2}} = \cdots = \frac{v_{B+1} - v_B}{v_B},$$

hence,

$$\frac{v_{i+1}}{v_i} = \left(\frac{M}{v_{l+1}}\right)^{\frac{1}{B-l}}, \; i = l+1, l+2, \ldots, B.$$

The above equations give the threshold values.
According to Lemma 4.3, we have that

$$\frac{v_{i+1} - v_i}{v_i} \frac{m}{v_{l+1}} l = 1, \; i = l+1, l+2, \ldots, B.$$

Thus, combining the above equations, we have

$$\frac{v_{l+1}}{m} = \left[\left(\frac{M}{v_{l+1}}\right)^{\frac{1}{B-l}} - 1\right] l.$$

Replacing v_{l+1} with $\frac{r^{(l)} m l}{B}$ yields Equation (7). □

We have introduced a threshold-based strategy OnAlg with a set of non-decreasing threshold values. The next theorem shows that OnAlg is optimal among all deterministic online algorithms.

Theorem 4.5. *The threshold-based algorithm* OnAlg *is optimal among all deterministic online algorithms.*

Proof. Let $\mathcal{A}$ be any deterministic online algorithm. In order to prove this theorem, we shall show that $\mathcal{A}$ cannot achieve a lower competitive ratio than that of OnAlg. Toward this, consider the following instance with increasing packet values

$$[\underbrace{(m) \times B}_{\text{first round}}, \underbrace{(m+\delta) \times B}_{\text{second round}}, \ldots, \underbrace{(m+(n-1)\delta) \times B}_{\text{penultimate round}}, \underbrace{(m+n\delta) \times B}_{\text{last round}}],$$

where $\delta = \frac{M-m}{n}$. By $\bar{v}_i$, we denote the packet value of the i-th packet that $\mathcal{A}$ admits under the above instance. In the first round, $\mathcal{A}$ is presented packets of packet value m for B times. If $\mathcal{A}$ never accept any packet, the adversary can stop the process right after the first round and construct an instance with the profit ratio being infinite. Assume $\mathcal{A}$ accepts $\bar{l} > 0$ packets in the first round. Let $\bar{B}$ be the total number of packets buffered during the above process. Then, the profit ratio between OPT and $\mathcal{A}$ when the adversary presents the entire instance to $\mathcal{A}$ is $\frac{MB}{\sum_{i=1}^{\bar{B}} \bar{v}_i}$.

Moreover, we assume during the above process, b-th ($b \geq \bar{l}$) and $(b+1)$-th packets are admitted in the j-th and j'-th round, respectively. If $j = j'$, the adversary can stop right after $\mathcal{A}$ admits the b-th packet. In this case, the profit obtained by $\mathcal{A}$ will be $\sum_{i=1}^{b} \bar{v}_i$, and that of OPT will not be less than $(\bar{v}_{b+1} - \delta)B$ (buffering packets during the $(j-1)$-th round). In this case, the profit ratio can be $\frac{(\bar{v}_{b+1}-\delta)B}{\sum_{i=1}^{b} \bar{v}_i}$. When $j \neq j'$, the adversary can stop the input instance right after the $(j'-1)$-th round and get a profit ratio of $\frac{(\bar{v}_{b+1}-\delta)B}{\sum_{i=1}^{b} \bar{v}_i}$. Thus, for any $\bar{l} \leq b \leq \bar{B} - 1$, we can always construct an instance under which the profit ratio between OPT and $\mathcal{A}$ is at least $\frac{(\bar{v}_{b+1}-\delta)B}{\sum_{i=1}^{b} \bar{v}_i}$.

With $\delta \to 0$, we have

$$\begin{aligned}
\textsc{cr}(\mathcal{A}) &\geq \max_{\bar{l} \leq b \leq \bar{B}} \left\{ \lim_{\delta \to 0} \frac{(\bar{v}_{b+1} - \delta)B}{\sum_{i=1}^{b} \bar{v}_i}, \frac{MB}{\sum_{i=1}^{\bar{B}} \bar{v}_i} \right\} \\
&\geq \max_{\bar{l} \leq b \leq \bar{B}} \left\{ \frac{\bar{v}_{b+1}B}{\sum_{i=1}^{b} \bar{v}_i}, \frac{MB}{\sum_{i=1}^{\bar{B}} \bar{v}_i} \right\} \\
&\geq \max_{\bar{l} \leq b \leq B} \frac{\bar{v}_{b+1}B}{\sum_{i=1}^{b} \bar{v}_i} \\
&\geq \min_{l, v_i} \max_{l \leq b \leq B} \frac{v_{b+1}B}{\sum_{i=1}^{b} v_i} = \textsc{cr}(\text{OnAlg}),
\end{aligned}$$

where in the penultimate inequality, we set $\bar{v}_i = \bar{v}_{\bar{B}}$ for $\bar{B} < x \leq B$ and $\bar{v}_{B+1} = M$.

This completes the proof. □

In [31], an online optimal solution to the online knapsack problem is derived by assuming that the item size is much smaller than the capacity of the knapsack. Using the solution in [31], a similar result can be obtained to problem (2) by allowing the number of packets to take fractional values. The following corollary restates this result simply for the convenience of the readers since it is required in our subsequent analysis for the general problem.

Corollary 4.6. *In the special case of allowing the number of packets to take fractional values, the optimal competitive ratio of Problem (2) is*

$$\textsc{cr}(\text{OnAlg}) = \ln \theta + 1,$$

and the optimal threshold function $g(x) : [0, B] \rightarrow [m, M]$ (which is a continuous analogue of the discrete threshold values) is

$$g(x) = \begin{cases} m, & x \leq \frac{B}{\ln \theta + 1}, \\ m e^{(\ln \theta + 1)\frac{x}{B} - 1}, & \text{otherwise.} \end{cases} \tag{8}$$

4.2 A Threshold-Based Strategy for the General Problem

In this section, we apply the *threshold-based* strategy to the original problem which allows packet departure, i.e., $u(t) \geq 0, t \in \mathcal{T}$. In order to analyze the performance of the *threshold-based* strategy, we can divide the investigated time period into multiple *cycles.*

Definition 2. *Given a threshold-based online algorithm* gOnAlg*, a cycle is defined to be the time interval beginning and ending whenever the buffer under* gOnAlg *becomes empty. Specifically, let $0 \leq t_1 < t_2 < \cdots < t_n \leq T$ denote the time slots when the queue length under* gOnAlg *goes to* 0*, then the time interval $[t_\tau + 1, t_{\tau+1}]$, $\tau \in \{1, 2, \ldots, n-1\}$ forms a cycle.*

The following observation implies that the analysis for the competitive ratio can be conducted within a *cycle.*

Lemma 4.7. *Let $\boldsymbol{\omega}$ be a worst instance and C be any cycle realized by* gOnAlg *under $\boldsymbol{\omega}$, then the profit ratio between* OPT *and* gOnAlg *during C is equal to the competitive ratio of* gOnAlg.

Proof. Assume under the worst instance $\boldsymbol{\omega} = [(v(\tau), u(\tau))]_{\tau \in \mathcal{T}}$, there is a cycle $C = [s+1, t]$ during which the maximum profit ratio is smaller than CR (gOnAlg). At time slots s and t, the buffer under gOnAlg is emptied. We just increase $u(s)$ and $u(t)$ by a large number λ such that the buffer under OPT also becomes empty. This operation never changes subsequent operations of gOnAlg, as well as the obtained profit. Also, the profit obtained by OPT is unchanged since $\boldsymbol{\omega}$ has resulted in the worst-case profit ratio. Then, we can "remove" the input segment over C and present the instance $[(v(\tau), u(\tau))]_{\tau=1:s-1} + [(v(s), u(s) + \lambda)] + [(v(\tau), u(\tau))]_{\tau=t+1:T}$ to gOnAlg and get a larger profit ratio, contradicting the assumption on the worst instance. Similarly, if the profit ratio during one cycle is larger than the competitive ratio, presenting the following input instance to gOnAlg yields the increase of the profit ratio:

$$[(v(\tau), u(\tau))]_{\tau=1:s-1} + [(v(s), u(s) + \lambda)] + \boldsymbol{\omega}_C \times 2 + [(v(\tau), u(\tau))]_{\tau=t+1:T},$$

where $\boldsymbol{\omega}_C = [(v(\tau), u(\tau))]_{\tau=s+1:t-1} + [(v(t), u(t) + \lambda)]$. This also contradicts the assumption on the worst instance. □

Based on the above lemma, our analysis on the competitive ratio of gOnAlg can be reduced to instances which only contain one cycle. By Ω_b^{gOnAlg}, we denote the subset of single-cycle input instances with the maximum queue length being b. The following lemma characterizes the local competitive ratio within such a subset.

Lemma 4.8. *Assume the threshold-based online algorithm* gOnAlg *is equipped with a series of non-decreasing threshold values v_i satisfying $v_1 = m$. If the length of the first step is l, we have:*

(1) *For the subset Ω_b^{gOnAlg} where $b < l$, the local competitive ratio is 1.*
(2) *For $b \geq l$, the local competitive ratio within the subset Ω_b^{gOnAlg} is*

$$\text{CR}_b(\text{gOnAlg}) \leq \frac{v_{b+1}B + \sum_{i=l+1}^{b} v_i}{\sum_{i=1}^{b} v_i}. \tag{9}$$

Inequality (9) holds with equality when the local competitive ratio within Ω_b^{gOnAlg} *satisfies* $\text{CR}_b(\text{gOnAlg}) \geq \text{CR}_{b'}(\text{gOnAlg})$ *for all* $b' < b$.

By Lemma 4.8, we have

$$\text{CR}(\text{gOnAlg}) = \max_{b=l,l+1,\ldots,B} \text{CR}_b(\text{gOnAlg}) = \max_{b=l,l+1,\ldots,B} \frac{v_{b+1}B + \sum\limits_{i=l+1}^{b} v_i}{\sum\limits_{i=1}^{b} v_i}.$$

Similar to Lemma 4.3, the next lemma explains how to optimize the threshold values v_i.

LEMMA 4.9. *If the length of the first step* l *is determined,* $\text{CR}(\text{gOnAlg}|l)$ *maximizes if and only if the following equalities hold:*

$$\frac{v_{l+1}B}{ml} = \frac{v_{l+2}B + v_{l+1}}{ml + v_{l+1}} = \cdots = \frac{v_B B + \sum\limits_{i=l+1}^{B-1} v_i}{ml + \sum\limits_{i=l+1}^{B-1} v_i} = \frac{MB + \sum\limits_{i=l+1}^{B} v_i}{ml + \sum\limits_{i=l+1}^{B} v_i}.$$

THEOREM 4.10. *Assuming* l *is given, the minimum competitive ratio of* gOnAlg, *denoted by* $r^{(l)}$ *satisfies*

$$\frac{B\theta}{r^{(l)}l} = \left(\frac{r^{(l)}(B-l)}{B^2} + 1\right)^{B-l}. \tag{10}$$

Assuming l^* *is the optimal solution to minimize the above function and* r^* *is the corresponding optimal value, then the optimal threshold values are*

$$v_{l^*+1} = \frac{r^* m l^*}{B},$$
$$v_{l^*+i+1} = v_{l^*+1}\left(\frac{M}{v_{l^*+1}}\right)^{\frac{i}{B-l^*}}, \quad i = 1, 2, \ldots, B - l^*.$$

PROOF. By induction, we can derive that

$$\frac{m}{v_{l+1}} \frac{Bl}{B-l} \frac{v_{i+1} - v_i}{v_i} = 1, \ i = l+1, l+2, \ldots, B. \tag{11}$$

That implies

$$\frac{v_{l+2} - v_{l+1}}{v_{l+1}} = \frac{v_{l+3} - v_{l+2}}{v_{l+2}} = \cdots = \frac{v_{B+1} - v_B}{v_B},$$

so

$$\frac{v_{i+1}}{v_i} = \left(\frac{M}{v_{l+1}}\right)^{\frac{1}{B-l}}, \quad i = l+1, l+2, \ldots, B. \tag{12}$$

Moreover, according to Lemma 4.9, we have

$$v_{l+1} = \frac{r^{(l)} ml}{B}.$$

Combining the above equation and (12), we can obtain the threshold values shown in the theorem. According to Equation (11), we have

$$\frac{v_{l+1}}{m} = \left[\left(\frac{M}{v_{l+1}}\right)^{\frac{1}{B-l}} - 1\right] \frac{Bl}{B-l}.$$

Replacing v_{l+1} with $\frac{r^{(l)}ml}{B}$ yields Equation (10). □

Remark 1. *By Equation (10), one can conclude that when B is large enough, the competitive ratio approximates $\frac{(2+\ln\theta)+\sqrt{\ln^2\theta+4\ln\theta}}{2}$, which is superior to the existing results ($e\lceil\ln\theta\rceil$ in [6] and $\ln\theta + 2 + O(\frac{\ln^2\theta}{B})$ in [5]).*

5 OPTIMAL RANDOMIZED ONLINE ALGORITHM

In the discrete case, the deterministic online algorithm can only admit or reject an incoming packet. It is beneficial for online algorithms to buffer a fraction of the packet if this is a viable option. Motivated by this observation, we focus on finding an optimal online algorithm for the fractional admission model. In this section, by considering fractional model, we design a novel online strategy and show it achieves the lower bound for competitive ratio of $\ln\theta + 1$. Then, in Section 5.2, we extend the result into the original discrete setting.

5.1 Optimal Online Algorithm for the Fractional Admission Model

In Section 4.1, we devised an algorithm for a simplified online problem without packet departure over the time horizon. We showed that for the discrete and fractional cases, the optimal competitive ratio can be obtained by a threshold-based algorithm, which maintains a fixed threshold value for each state of the queue length. For the general case with packet departure, the available space for buffering packets changes over the time. Thus, the threshold-based strategy whose threshold only depends on the queue length may yield a suboptimal competitive ratio. In this section, we aim to find the optimal competitive ratio for the general case. Toward this goal, we propose a novel online algorithm fOnAlg. In a nutshell, we introduce virtual sub-queues which track the history of packet departures and maintain a threshold-based strategy for each virtual sub-queue. The details of fOnAlg are as follows.

5.1.1 State of fOnAlg. Without loss of generality, we assume the initial state of the queue length is 0. fOnAlg initially sets B empty sub-queues, each of which with capacity 1. This corresponds to the initial buffer budget of B. Note that packet departure may or may not occur at any time slot. Hence, let t_i, $i = 1, 2, \ldots, h$ denote the time slots that at least one packet departure occurs. If there are $u(t_i)$ packets departure in slot t_i, then $u(t_i)$ virtual sub-queues, each with capacity 1, will be created by the end of slot t_i. When the original queue length $b(t)$ becomes 0, the virtual sub-queues created due to packet departures is deleted and the queue lengths of the initial B sub-queues are set to 0. Let D be a sufficiently large integer value. A packet is conceptually divided into D equal *sub-packets*. We denote the discretized queue length of the i-th sub-queue by $\bar{b}_i(t)$ which can only take integer values between 0 and D. Using the above setups, we define fOnAlg algorithm by extending the original system state $b(t)$ to a state vector $\bar{\mathbf{b}}(t) = [\bar{b}_1(t), \bar{b}_2(t), \ldots, \bar{b}_n(t)]$, where n is the number of virtual sub-queues at time slot t. It is easy to see that $b(t)$ equals to $\sum_{i=1}^{n} \bar{b}_i(t) - n + B$. Figure 2 depicts the state of fOnAlg at time slot t.

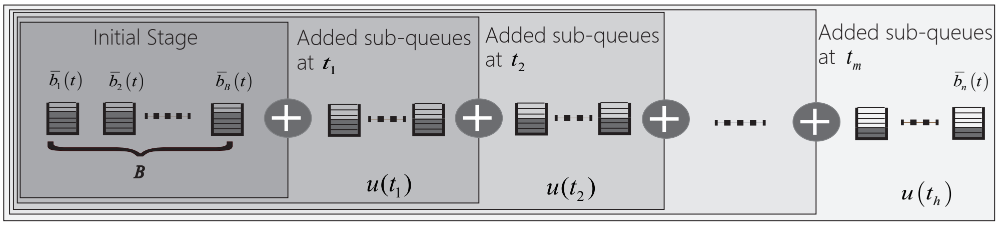

Fig. 2. Description of the state of fOnAlg at time slot t.

5.1.2 Admission Policy of fOnAlg. In fOnAlg, admission policy determines the fraction of the packet to be admitted, i.e., the number of sub-packets in $[0, D]$ that must be admitted. The number of admitted sub-packet depends on the system state $\bar{\mathbf{b}}(t)$. Toward this, for each sub-queue i, fOnAlg maintains a fixed threshold function $g_D(\bar{b}_i)$ that can be used to determine the number of admitted sub-packets of an arriving packet. The function is defined by

$$g_D(\bar{b}_i) = \begin{cases} m, & \frac{\bar{b}_i}{D} \leq \frac{1}{\ln\theta+1}, \\ m e^{(\ln\theta+1)\frac{\bar{b}_i}{D}-1}, & \text{otherwise}. \end{cases} \tag{13}$$

Equation (13) is a scaled version of (8). Based on Equation (13), the number of sub-packets that the i-th sub-queue can admit, denoted by $\bar{x}_i(t)$ is determined by

$$\begin{aligned} \bar{x}_i(t) = \quad & \max \quad \epsilon \\ & \text{s.t.} \quad g_D(\bar{b}_i(t-1)+\epsilon) \leq p(t), \\ & \text{var.} \quad \epsilon \in \mathbb{N}. \end{aligned}$$

By the above admission policy, the value of admitted sub-packets is guaranteed to be larger than the threshold of each sub-queue at any time slot. Then, the total number of aggregated sub-packets is simply the aggregation of sub-packets in each sub-queue truncated by D, i.e.,

$$\bar{x}(t) = \min\left\{\sum_{i=1}^{n} \bar{x}_i(t), D\right\}.$$

The value of $\bar{x}(t)$ is the admitted amount of an arriving packet by fOnAlg under state $\bar{\mathbf{b}}(t-1)$ and packet value $v(t)$. We use $\bar{g}_D(\bar{\mathbf{b}})$ to denote the minimum threshold for fOnAlg to admit portion of packet when the algorithm state is $\bar{\mathbf{b}}$, i.e.,

$$\bar{g}_D(\bar{\mathbf{b}}) = \min_{i=1,2,\cdots,n} g_D(\bar{b}_i+1).$$

$\bar{g}_D(\bar{\mathbf{b}})$ only depends on the algorithm state $\bar{\mathbf{b}}$.

5.1.3 Allocation Policy of fOnAlg. Given that the number of sub-packets to be buffered is $\bar{x}(t)$, fOnAlg allocates these sub-packets among the n sub-queues in a *water-filling* way (as shown in Figure 3). In this way, the value of an allocated sub-packet is always larger than the threshold value of the sub-queue. Specifically, the allocation algorithm is as follows.

ALGORITHM 1: Allocation Policy of fOnAlg (Water-filling policy)

```
for $i = 1$ to $\bar{x}(t)$ do
    $\mathcal{I} \leftarrow$ The sub-queues with smallest queue length
    Allocate $i$-th sub-packet to the sub-queue with the smallest index in $\mathcal{I}$, and increase its queue length by 1.
```

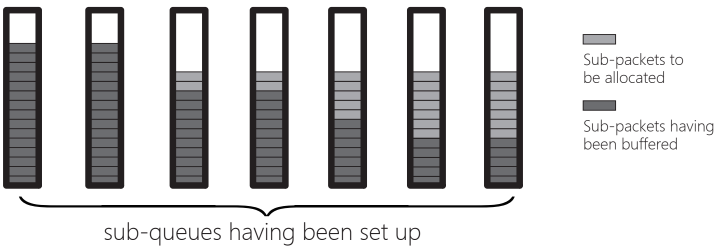
Sub-packets to be allocated
Sub-packets having been buffered
sub-queues having been set up

Fig. 3. Allocate admitted sub-packets in a *water-filling* manner.

5.1.4 Performance Analysis of fOnAlg. When a packet departs, a unit of buffer space is released and the available buffering budget increases accordingly. The conceptual idea of fOnAlg is to build a virtual sub-queue for each unit of released buffering space. An arriving packet is broken into multiple sub-packets and admitted to the original buffer and sub-queues under a thresholding policy such that that the packet value of a buffered sub-packet is not less than the threshold. In order to analyze the competitive ratio of fOnAlg, we face fOnAlg with a more "powerful" adversary that can transmit a portion of packet aiming to construct a worse instance. Let $d(t),\ 1 \leq d(t) \leq D$ denotes the number of sub-packets transmitted to the switch node at time slot t. Under the powerful adversary, an input instance has the following extended form

$$[\bar{\omega}(t) = (v(t), u(t), d(t))]_{t\in\mathcal{T}},$$

where $v(t)$ and $u(t)$ are as the previous exogenous inputs. Note that when $d(t) = D,\ t \in \mathcal{T}$, the input instance will be reduced to the previous one defined in Section 3. Thus, the competitive ratio of fOnAlg when faced with the powerful adversary characterizes an upper bound for the competitive ratio of fOnAlg (under a standard adversary defined in Section 3).

By analyzing the admission and allocation policies of fOnAlg, we observe the following two critical properties:

(1) When queue lengths are equal, fOnAlg will allocate the sub-packet to the sub-queue with smaller index. Thus, at any slot t, the following inequalities always hold:
$$\bar{b}_1(t) \geq \bar{b}_2(t) \geq \cdots \geq \bar{b}_n(t).$$
(2) When a sub-packet is allocated to a sub-queue, the packet value is always larger than or equal to the threshold value of the sub-queue.

Under the powerful adversary, the following Lemma characterizes a critical property of the worst instance for fOnAlg.

Lemma 5.1. *Assume $\bar{\boldsymbol{\omega}} = [\bar{\omega}(t) = (v(t), u(t), d(t))]_{t\in\mathcal{T}}$ is a worst instance for* fOnAlg *under the powerful adversary. Then, at any time slot t when* fOnAlg *buffers portion of packet, $v(t)$ is exactly equal to the threshold of* fOnAlg, *i.e.,*

$$v(t) = \bar{g}_D(\bar{\mathbf{b}}(t-1)).$$

Proof. Assume during the worst instance, there exists a time slot that $v(t)$ is larger than the threshold value $\bar{g}_D(\bar{\mathbf{b}}(t-1))$ and $\bar{x}(t)$ sub-packets are admitted by fOnAlg. We first present the instance segment $[(v(1), u(1), d(1)), \ldots, (v(t-1), u(t-1), d(t-1))]$ to fOnAlg. After that, we further present the instance segment $[(\bar{g}_D(\bar{\mathbf{b}}(t-1)), 0, 1), (v(t), 0, \bar{x}(t)-1)]$ to fOnAlg, and the number of buffered sub-packets is equal to $\bar{x}(t)$. In this way, the profit earned from the buffered $\bar{x}(t)$ sub-packets decreases by $[v(t) - \bar{g}_D(\bar{\mathbf{b}}(t-1))]/D$. After that, we present the remaining instance segment $[(v(t+1), u(t+1), d(t+1)), (v(t+2), u(t+2), d(t+2)), \ldots, (v(T), u(T), d(T))]$ to fOnAlg. Under the above instance, the profit earned by OPT decreases by at most $[v(t) - \bar{g}_D(\bar{\mathbf{b}}(t-1))]/D$, which is also the decrement amount of fOnAlg. In this way, we construct a new instance that results in a larger competitive ratio, contradicting the assumption on the worst instance. □

The next Lemma implies that the competitive analysis of fOnAlg can be conducted within a *cycle* (see Definition 2).

Lemma 5.2. *Let $\bar{\boldsymbol{\omega}}$ be a worst instance and C be a cycle realized by* fOnAlg *under $\bar{\boldsymbol{\omega}}$, then the profit ratio during C is equal to the competitive ratio of* fOnAlg.

The proof is analogous to that for Lemma 4.7 and is omitted. The following theorem shows the competitive ratio of fOnAlg.

THEOREM 5.3. *Under the fractional admission assumption,* fOnAlg *achieves the competitive ratio* $\left[1 + (\ln\theta + 1)\frac{\theta}{D}\right] \cdot (\ln\theta + 1)$.

PROOF. We assume there is a worst instance for fOnAlg which only contains one cycle and at time t, the state of fOnAlg is as shown in Figure 2. By s_i, we denote the time slot at which the i-th sub-queue is built up. From the allocation policy of fOnAlg, we have

$$\bar{b}_1(t) \geq \bar{b}_2(t) \geq \cdots \geq \bar{b}_n(t).$$

During $[s_i + 1, t]$, the threshold values of fOnAlg is always less than or equal to $g_D(\bar{b}_i(t) + 1)$. Thus, during the worst instance, packet values in $[s_i + 1, t]$ are always less than or equal to $g_D(\bar{b}_i(t) + 1)$. Otherwise, the packet value will be larger than the threshold of fOnAlg, contradicting Lemma 5.1.

Above analysis demonstrates that under the worst instance for fOnAlg, packet values are not larger than $g_D(\bar{b}_i(t) + 1)$ after the i-th sub-queue is built up. Thus, the profit earned by OPT over $[1, t]$ is less than or equal to

$$\mathsf{Prof}_{\mathsf{OPT}} \leq g_D(\bar{b}_1(t) + 1) + g_D(\bar{b}_2(t) + 1) + \cdots + g_D(\bar{b}_n(t) + 1). \tag{14}$$

On the other hand, according to the admission policy of fOnAlg, a sub-packet is admitted by a sub-queue only when its packet value is larger than or equal to the threshold, so the aggregated profit of the buffered sub-packets in the i-th sub-queue is not less than $\sum_{k=1}^{\bar{b}_i(t)} g_D(k)$. By the property of the threshold function g_D (see the analysis in Section 4.1), we have

$$\sum_{k=1}^{\bar{b}_i(t)} g_D\left(k\right) \geq \frac{g_D\left(\bar{b}_i(t)\right)}{\ln\theta + 1}.$$

Hence, the profit earned by fOnAlg over $[1, t]$ is larger than or equal to

$$\mathsf{Prof}_{\mathsf{fOnAlg}} \geq \frac{g_D(\bar{b}_1(t)) + g_D(\bar{b}_2(t)) + \cdots + g_D(\bar{b}_n(t))}{\ln\theta + 1}. \tag{15}$$

Putting together Equations (14) and (15), we have

$$\begin{aligned}
\textsc{cr}(\mathsf{fOnAlg}) &\leq \frac{g_D(\bar{b}_1(t) + 1) + g_D(\bar{b}_2(t) + 1) + \cdots + g_D(\bar{b}_n(t) + 1)}{g_D(\bar{b}_1(t)) + g_D(\bar{b}_2(t)) + \cdots + g_D(\bar{b}_n(t))} \cdot (\ln\theta + 1) \\
&\leq \max_{1\leq i\leq n} \frac{g_D(\bar{b}_i(t) + 1)}{g_D(\bar{b}_i(t))} \cdot (\ln\theta + 1) \\
&\leq \left[1 + (\ln\theta + 1)\frac{\theta}{D}\right] \cdot (\ln\theta + 1).
\end{aligned}$$

The last inequality is according to the fact that the largest derivative of $g_D(\bar{b}_i)$, $0 \leq \bar{b}_i \leq D$ is $m(\ln\theta + 1)\theta\frac{1}{D}$ and $g(\bar{b}_i) \geq m$. □

COROLLARY 5.4. *Assuming $D \to \infty$, the competitive ratio of* fOnAlg *approximates the lower bound of* $\ln\theta + 1$.

REMARK 2. *We note the following property concerning the admission amount when $D \to \infty$. For notational convenience, let us denote by $\tilde{b}_i(t)$ the actual amount of packet in the i-th sub-queue and use $\tilde{x}_i(t)$ to denote the actual amount of packet that can be admitted by the i-th sub-queue. According to the admission policy of* fOnAlg, *we can deduce that when $D \to \infty$, the amount of packet that the i-th sub-queue can admit is*

$$\tilde{x}_i(t) = \left[\frac{\ln\frac{v(t)}{m} + 1}{\ln\theta + 1} - \tilde{b}_i(t-1)\right]^+ .$$

Similarly, the actual admitted amount at time slot t, denoted by $x^{\text{fOnAlg}}(t)$, can be obtained by truncating the aggregation of $\tilde{x}_i(t)$ for all sub-queues. That is,

$$x^{\text{fOnAlg}}(t) = \min\left\{\sum_{i=1}^{n} \tilde{x}_i(t), 1\right\}.$$

Then, a fraction of $x^{\text{fOnAlg}}(t)$ of the packet will be allocated among sub-queues in a water-filling way as shown in Figure 3.

5.2 Optimal Randomized Strategy for the Discrete Admission Model

In this section, we extend the fractional algorithm in the previous section and design a randomized online algorithm for the general discrete model. Intuitively, the expected competitive ratio can be improved to be equal to that of fOnAlg by properly designing the buffering probability. In the following, we introduce a randomized strategy rOnAlg (as for randomized online algorithm) which can attain this goal by online rounding. Given that the amount of packet buffered by fOnAlg assuming $D \to \infty$ at time slot t is $x^{\text{fOnAlg}}(t)$, we propose the following operation rules for rOnAlg.

At time slot t, our proposed randomized rounding algorithm computes the admission probability according to the admission amount of fOnAlg with $D \to \infty$ (summarized as Algorithm 2).

ALGORITHM 2: Randomized Rounding Algorithm rOnAlg, at slot t

```
$q(t) \leftarrow 0$ the admission probability of packet at slot $t$
if $\lfloor b^{\text{fOnAlg}}(t-1) + x^{\text{fOnAlg}}(t) \rfloor = \lfloor b^{\text{fOnAlg}}(t-1) \rfloor$ then
    if $b^{\text{rOnAlg}}(t-1) = \lfloor b^{\text{fOnAlg}}(t-1) \rfloor$ then
        $q(t) \leftarrow \frac{x^{\text{fOnAlg}}(t)}{\lfloor b^{\text{fOnAlg}}(t-1) \rfloor + 1 - b^{\text{fOnAlg}}(t-1)}$        (16)
    else
        $q(t) \leftarrow 0$
if $\lfloor b^{\text{fOnAlg}}(t-1) + x^{\text{fOnAlg}}(t) \rfloor = \lfloor b^{\text{fOnAlg}}(t-1) \rfloor + 1$ then
    if $b^{\text{rOnAlg}}(t-1) = \lfloor b^{\text{fOnAlg}}(t-1) \rfloor$ then
        $q(t) \leftarrow 1$
    else
        $q(t) \leftarrow \frac{x^{\text{fOnAlg}}(t) + b^{\text{fOnAlg}}(t-1) - \lfloor b^{\text{fOnAlg}}(t-1) \rfloor - 1}{b^{\text{fOnAlg}}(t-1) - \lfloor b^{\text{fOnAlg}}(t-1) \rfloor}$        (17)
Admit the packet with probability $q(t)$
```

Before analyzing the competitive ratio of above randomized policy, we first provide a general lower bound for the competitive ratio of any randomized online algorithms. Subsequent analysis demonstrates that the obtained lower bound is tight.

Theorem 5.5. *The competitive ratio for any randomized online algorithm is lower bounded by* $\ln \theta + 1$.

Proof. Given the initial state of the buffer being empty, an adversary can present enough packets with packet value $\bar{v}_1 = m$ to a randomized online algorithm $\mathcal{R}$. It is obvious that the number of packets admitted by $\mathcal{R}$ will increase and definitely converge to some constant value, denoted by l_1. Note that l_1 can be regarded as a random variable for a randomized online algorithm. After that, we further present enough packets with packet value $\bar{v}_2 > \bar{v}_1$ to $\mathcal{R}$, and assume ultimately there are l_2 packets buffered. We repeat this process for n times and let $\bar{v}_n = M$. Since the adversary can choose to stop the above input instance at any time. Hence, for such an input instance, it is possible for any

$\mathcal{R}$ to achieve a profit ratio equal to $(\bar{v}_i B)/\left(\sum_{k=1}^{i} \bar{v}_k l_k\right)$, $i = 1, 2, \ldots, n$. The expected competitive ratio is larger than or equal to the maximum one among those ratios. That is

$$\textsc{ecr}(\mathcal{R}) \geq \max_{i=1,2,\ldots,n} \frac{\bar{v}_k B}{\boldsymbol{E}\left[\sum\limits_{k=1}^{i} \bar{v}_k l_k\right]} = \max_{i=1,2,\ldots,n} \frac{\bar{v}_k B}{\sum\limits_{k=1}^{i} \bar{v}_k \boldsymbol{E}\left[l_k\right]}.$$

Obviously, $\sum\limits_{k=1}^{i} \boldsymbol{E}\left[l_k\right] \leq B$. From the previous analysis, we can verify that

$$\max_{i=1,2,\ldots,n} \frac{\bar{v}_i B}{\sum_{k=1}^{i} \bar{v}_k \boldsymbol{E}\left[l_k\right]} \geq \ln\theta + 1,$$

and thus $\textsc{ecr}(\mathcal{R}) \geq \ln\theta + 1$, the proof is completed. □

Lemma 5.6. $\left\lfloor b^{\text{fOnAlg}}(t) \right\rfloor \leq b^{\text{rOnAlg}}(t) \leq \left\lfloor b^{\text{fOnAlg}}(t) \right\rfloor + 1.$

Proof. Assuming the initial state of the buffer is empty, we proceed to prove by induction. For the base case we have

$$\left\lfloor b^{\text{fOnAlg}}(t) \right\rfloor \leq b^{\text{rOnAlg}}(t) \leq \left\lfloor b^{\text{fOnAlg}}(t) \right\rfloor + 1,$$

which holds when $t = 0$.

Now, assume the lemma holds for $t \leq k$, i.e., $b^{\text{rOnAlg}}(t)$ is equal to either $\left\lfloor b^{\text{fOnAlg}}(k) \right\rfloor$ or $\left\lfloor b^{\text{fOnAlg}}(k) \right\rfloor + 1$. As illustrated in Figure 4, when $b^{\text{rOnAlg}}(t)$ lies at point "A", the arriving packet is buffered with probability 0; and when $b^{\text{rOnAlg}}(t)$ lies at point "B", the arriving packet is buffered with probability 1.

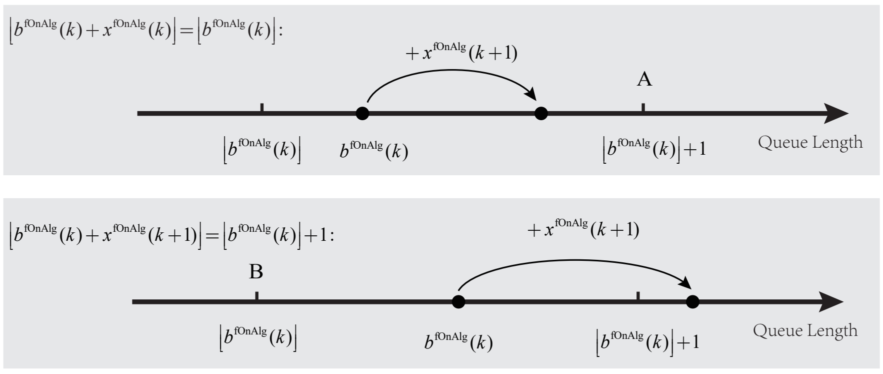
$\lfloor b^{\text{fOnAlg}}(k)+x^{\text{fOnAlg}}(k)\rfloor=\lfloor b^{\text{fOnAlg}}(k)\rfloor$:
$+x^{\text{fOnAlg}}(k+1)$
A
$\lfloor b^{\text{fOnAlg}}(k)\rfloor$
$b^{\text{fOnAlg}}(k)$
$\lfloor b^{\text{fOnAlg}}(k)\rfloor+1$
Queue Length
$\lfloor b^{\text{fOnAlg}}(k)+x^{\text{fOnAlg}}(k+1)\rfloor=\lfloor b^{\text{fOnAlg}}(k)\rfloor+1$:
$+x^{\text{fOnAlg}}(k+1)$
B
$\lfloor b^{\text{fOnAlg}}(k)\rfloor$
$b^{\text{fOnAlg}}(k)$
$\lfloor b^{\text{fOnAlg}}(k)\rfloor+1$
Queue Length

Fig. 4. The illustration of the proof for Lemma 5.6.

Putting together the observations in Figure 4, the following inequalities hold:

$$\begin{aligned}\left\lfloor b^{\text{fOnAlg}}(k) + x^{\text{fOnAlg}}(k+1) \right\rfloor &\leq b^{\text{rOnAlg}}(k) + x^{\text{rOnAlg}}(k+1) \\ &\leq \left\lfloor b^{\text{fOnAlg}}(k) + x^{\text{fOnAlg}}(k+1) \right\rfloor + 1.\end{aligned}$$

Hence,

$$\begin{aligned}\left[\left\lfloor b^{\text{fOnAlg}}(k) + x^{\text{fOnAlg}}(k+1) \right\rfloor - u(k+1)\right]^+ &\leq \left[b^{\text{rOnAlg}}(k) + x^{\text{rOnAlg}}(k+1) - u(k+1)\right]^+ \\ &\leq \left[\left\lfloor b^{\text{fOnAlg}}(k) + x^{\text{fOnAlg}}(k+1) \right\rfloor + 1 - u(k+1)\right]^+,\end{aligned}$$

for $u(k+1) \in \mathbb{N}$, so

$$\left\lfloor b^{\text{fOnAlg}}(k+1) \right\rfloor \leq b^{\text{rOnAlg}}(k+1) \leq \left\lfloor b^{\text{fOnAlg}}(k+1) \right\rfloor + 1.$$

This completes the proof. □

Lemma 5.7. $\boldsymbol{E}\left[x^{\text{rOnAlg}}(t)\right] = x^{\text{fOnAlg}}(t)$.

Proof. We prove by induction again. Assume the initial state of the queue length is 0, i.e., $b^{\text{fOnAlg}}(0) = b^{\text{rOnAlg}}(0) = 0$. If $x^{\text{fOnAlg}}(1) < 1$, then

$$\left\lfloor b^{\text{fOnAlg}}(0) + x^{\text{fOnAlg}}(1) \right\rfloor = \left\lfloor b^{\text{fOnAlg}}(0) \right\rfloor.$$

Hence, according to Equation (16), the packet is admitted with probability

$$\frac{x^{\text{fOnAlg}}(1)}{\left\lfloor b^{\text{fOnAlg}}(0) \right\rfloor + 1 - b^{\text{fOnAlg}}(0)} = x^{\text{fOnAlg}}(1).$$

If $x^{\text{fOnAlg}}(1) = 1$, then

$$\left\lfloor b^{\text{fOnAlg}}(0) + x^{\text{fOnAlg}}(1) \right\rfloor = \left\lfloor b^{\text{fOnAlg}}(0) \right\rfloor + 1.$$

The arriving packet will be admitted with probability 1. So when $t = 1$, we have $\boldsymbol{E}\left[x^{\text{rOnAlg}}(t)\right] = x^{\text{fOnAlg}}(t)$ and also $\boldsymbol{E}\left[b^{\text{rOnAlg}}(t)\right] = b^{\text{fOnAlg}}(t)$.

Now, assume $\boldsymbol{E}\left[x^{\text{rOnAlg}}(t)\right] = x^{\text{fOnAlg}}(t)$ and $\boldsymbol{E}\left[b^{\text{rOnAlg}}(t)\right] = b^{\text{fOnAlg}}(t)$ hold for $t \leq k$. Based on Lemma 5.6, we know that $b^{\text{rOnAlg}}(t)$ is equal to either $\left\lfloor b^{\text{fOnAlg}}(t) \right\rfloor$ or $\left\lfloor b^{\text{fOnAlg}}(t) \right\rfloor + 1$. When $\boldsymbol{E}\left[b^{\text{rOnAlg}}(k)\right] = b^{\text{fOnAlg}}(k)$, we can deduce that $b^{\text{rOnAlg}}(k)$ is equal to $\left\lfloor b^{\text{fOnAlg}}(k) \right\rfloor$ with probability $\left\lfloor b^{\text{fOnAlg}}(k) \right\rfloor + 1 - b^{\text{fOnAlg}}(k)$ and equal to $\left\lfloor b^{\text{fOnAlg}}(k) \right\rfloor + 1$ with probability $b^{\text{fOnAlg}}(k) - \left\lfloor b^{\text{fOnAlg}}(k) \right\rfloor$. By verifying the operations of rOnAlg, we can easily get that $\boldsymbol{E}\left[x^{\text{rOnAlg}}(k+1)\right] = x^{\text{fOnAlg}}(k+1)$. Moreover, if $b^{\text{fOnAlg}}(k) + x^{\text{fOnAlg}}(k+1) \geq u(k+1)$, we must have $b^{\text{rOnAlg}}(k) + x^{\text{rOnAlg}}(k+1) \geq u(k+1)$ according to Lemma 5.6. In this case, we have

$$\begin{aligned}
\boldsymbol{E}\left[b^{\text{rOnAlg}}(k+1)\right] =& \boldsymbol{E}\left[\left[b^{\text{rOnAlg}}(k) + x^{\text{rOnAlg}}(k+1) - u(k+1)\right]^{+}\right] \\
=& \boldsymbol{E}\left[b^{\text{rOnAlg}}(k) + x^{\text{rOnAlg}}(k+1) - u(k+1)\right] \\
=& \boldsymbol{E}\left[b^{\text{rOnAlg}}(k)\right] + \boldsymbol{E}\left[x^{\text{rOnAlg}}(k+1)\right] - u(k+1) \\
=& b^{\text{fOnAlg}}(k+1).
\end{aligned}$$

If $b^{\text{fOnAlg}}(k) + x^{\text{fOnAlg}}(k+1) \leq u(k+1)$, we must have $b^{\text{rOnAlg}}(k) + x^{\text{rOnAlg}}(k+1) \leq u(k+1)$. In this case, $b^{\text{rOnAlg}}(k+1) = b^{\text{fOnAlg}}(k+1) = 0$. Then, we can conclude that $\boldsymbol{E}\left[x^{\text{rOnAlg}}(t)\right] = x^{\text{fOnAlg}}(t)$ and $\boldsymbol{E}\left[b^{\text{rOnAlg}}(t)\right] = b^{\text{fOnAlg}}(t)$ hold for $t = k+1$. By concluding the above all, we can get the final results as desired. □

Theorem 5.8. rOnAlg *achieves the optimal competitive ratio, i.e.,* $\ln\theta + 1$.

Proof. As shown in Corollary 5.4, when $D \to \infty$, the competitive ratio of fOnAlg achieves $\ln\theta + 1$. Moreover, from Lemma 5.7, we have that the expected amount of admitted packet under rOnAlg is always equal to that of fOnAlg for all times, and thus rOnAlg achieves the same competitive ratio as fOnAlg with $D \to \infty$. This proves the optimality of rOnAlg. □

6 CONCLUSION

In this paper, we studied the classic non-preemptive QoS buffer management problem, where its optimal online solution was an open problem for a decade. By relaxing the original discrete model to a fractional setting, we proposed a novel online algorithm that maintains a series of virtual sub-queues to record the available buffer budget over time. We proved it achieves the optimal competitive ratio. By devising a randomized rounding scheme, we then extended the fractional algorithm into the original discrete setting. Our analysis demonstrated that the randomized scheme achieves the optimal competitive ratio of $\ln \theta + 1$.

Last but not the least, the problem is of interest in a general admission control problem that could be applied to the state-of-the-art applications. As an example, we stress the value-based resource allocation in data centers with limited computation capacities, in which the jobs must be either admitted or rejected upon their arrival based on their values and the utilization of the servers.

7 ACKNOWLEDGMENT

This work was supported in part by a grant from the Research Grants Council of the Hong Kong Special Administrative Region under Project GRF 14630915.

REFERENCES

[1] W Aiello, Y Mansour, S Rajagopolan, and A Rosén. 2000. Competitive queue policies for differentiated services. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. 431–440.

[2] M Ajtai, N Megiddo, and O Waarts. 2001. Improved algorithms and analysis for secretary problems and generalizations. *SIAM Journal on Discrete Mathematics* 14, 1 (2001), 1–27.

[3] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. 2013. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing* 6, 5 (2013), 93–106.

[4] N. Andelman. 2005. Randomized qeue management for DiffServ. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 1–10.

[5] N. Andelman and Y. Mansour. 2003. Competitive management of non-preemptive queues with multiple values. In *Proceedings of the International Symposium on Distributed Computing (DISC)*. 166–180.

[6] N. Andelman, Y. Mansour, and A. Zhu. 2003. Competitive queueing policies for QoS switches. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 761–770.

[7] Y. Azar and Y. Richter. 2005. Management of multi-queue switches in QoS networks. *Algorithmica* 43, 1-2 (2005), 81–96.

[8] M Babaioff, N Immorlica, D Kempe, and R Kleinberg. 2007. A knapsack secretary problem with applications. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* (2007), 16–28.

[9] H Böckenhauer, D Komm, R Královič, and P Rossmanith. 2014. The online knapsack problem: Advice and randomization. *Theoretical Computer Science* 527 (2014), 61–72.

[10] A. Borodin and R El-Yaniv. 1998. *Online computation and competitive analysis.* Cambridge University Press.

[11] P. Chuprikov, S. Nikolenko, and K. Kogan. 2015. Priority queueing with multiple packet characteristics. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. 1418–1426.

[12] R El-Yaniv, A Fiat, R Karp, and G Turpin. 1992. Competitive analysis of financial games. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*. 327–333.

[13] R. El-Yaniv, A. Fiat, R. M. Karp, and G Turpin. 2001. Optimal search and one-way trading online algorithms. *Algorithmica* 30, 1 (2001), 101–139.

[14] M Englert and Westermann M. 2012. Considering suppressed packets improves buffer management in quality of service switches. *SIAM J. Comput.* 41, 5 (2012), 1166–1192.

[15] M. Englert and M. Westermann. 2009. Lower and upper bounds on FIFO buffer management in QoS switches. *Algorithmica* 53, 4 (2009), 523–548.

[16] Moran Feldman and Joseph Seffi Naor. 2017. Non-preemptive buffer management for latency sensitive packets. *Journal of Scheduling* 20, 4 (2017), 337–353.

[17] A Fiat, Y Mansour, H Yi, and U Nadav. 2008. Competitive queue management for latency sensitive packets. In *Proceedings of ACM-SIAM Symposium on Discrete algorithms (SODA)*. 228–237.

[18] P R Freeman. 1983. The secretary problem and its extensions: A review. *International Statistical Review* (1983), 189–206.

[19] M. Goldwasser. 2010. A survey of buffer management policies for packet switches. *ACM SIGACT News* 41, 1 (2010), 100–128.

[20] A Kesselman, Z Lotker, Y Mansour, B Patt-Shamir, and B Schieber. 2004. Buffer overflow management in QoS switches. *SIAM J. Comput.* 33, 3 (2004), 563–583.

[21] A. Kesselman and Y. Mansour. 2001. Loss-bounded analysis for differentiated services. In *Proceedings of ACM-SIAM Symposium on Discrete algorithms (SODA)*. 591–600.

[22] K. Kobayashi, S. Miyazaki, and Y. Okabe. 2009. Competitive buffer management for multi-queue switches in QoS networks using packet buffering algorithms. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 328–336.

[23] J. Lorenz, K. Panagiotou, and A Steger. 2009. Optimal algorithms for k-search with application in option pricing. *Algorithmica* 55, 2 (2009), 311–328.

[24] Z. Lotker and B. Patt-Shamir. 2002. Nearly optimal FIFO buffer management for DiffServ. In *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC)*. 134–143.

[25] Y. Mansour, B. Patt-Shamir, and O. Lapid. 2000. Optimal smoothing schedules for real-time streams. In *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC)*. 21–29.

[26] S Moharir, S Sanghavi, and S Shakkottai. 2013. Online load balancing under graph constraints. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 41. 363–364.

[27] E. Mohr, I. Ahmad, and G. Schmidt. 2014. Online algorithms for conversion problems: a survey. *Surveys in Operations Research and Management Science* 19, 2 (2014), 87–104.

[28] W Shi, L Zhang, C Wu, Z Li, and F Lau. 2014. An online auction framework for dynamic resource provisioning in cloud computing. *ACM SIGMETRICS Performance Evaluation Review* 42, 1 (2014), 71–83.

[29] L Yang, M H Hajiesmaili, H Yi, and M Chen. 2017. Hour-Ahead Offering Strategies in Electricity Market for Power Producers with Storage and Intermittent Supply. In *Proceedings of ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*. 21–22.

[30] Zijun Zhang, Zongpeng Li, and Chuan Wu. 2017. Optimal Posted Prices for Online Cloud Resource Allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 1 (2017), 23.

[31] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In *International Workshop on Internet and Network Economics*. 566–576.

[32] A. Zhu. 2004. Analysis of queueing policies in QoS switches. *Journal of Algorithms* 53, 2 (2004), 137–168.

A PROOFS

A.1 Proof of Lemma 4.3

Proof. By partially dualizing on the first set of nonequality constraints, we obtain the following Lagrangian function:

$$L(y, \boldsymbol{v}, \boldsymbol{\mu}) = y + \sum_{i=0}^{B-l} \mu_i (v_{l+i+1}B - y \sum_{j=1}^{l+i} v_j),$$

where $\mu_i,\ i = 0, 1, \dots, B-l$ are the dual variables associated with the non-equality constraints.

The first-order optimality necessity condition can be expressed as

$$1 - \sum_{i=0}^{B-l} \mu_i \sum_{j=1}^{l+i} v_j = 0,$$

$$\mu_i B - y \sum_{k=i+1}^{B-l} \mu_k = 0,\ i = 0, 1, \dots, B-l-1.$$

Note that $m \le v_j \le M$ for $j = l+1, l+2, \dots, B-1$. Then, by the means of above equations, we can show that $\mu_i > 0,\ i = 0, 1, \dots, B-l$. Moreover, according to the complementary slackness condition,

$$\mu_i (v_{l+i+1}B - y \sum_{j=1}^{l+i} v_j) = 0,\ i = 0, 1, \dots, B-l.$$

This implies that the following equations always hold:

$$v_{l+i+1}B - y\sum_{j=1}^{l+i} v_j = 0,\ i = 0, 1, \ldots, B - l.$$

Mathematically, this can be rewritten as

$$\frac{v_{l+i+1}B}{\sum\limits_{j=1}^{l+i} v_j} = y,\ i = 0, 1, \ldots, B - l.$$

This completes the proof. □

A.2 Proof of Corollary 4.6

Proof. For the fractional case, a packet can be equally divided into arbitrarily small units. Here n is a large number. Specifically, we divide a packet into n units. Let the *threshold-based* policy set a threshold value for each unit of packet. Then by optimizing the threshold values and setting the first step length $l = \alpha B$, we attain a competitive ratio r satisfying

$$\left(\frac{r+Bn}{Bn}\right)^{Bn-\alpha Bn+1} - \left(\frac{r+Bn}{Bn}\right)^{Bn-\alpha Bn} - \frac{\theta}{\alpha Bn} = 0.$$

Let $n \to \infty$, we have

$$e^{r(1-\alpha)} = \frac{\theta}{\alpha r}. \tag{18}$$

Equation (18) defines an implicit function of r with respect to α. By taking derivative, we have that when $\alpha = 1/r$, r takes the minimum value. Substituting $\alpha = 1/r$ to Equation (18) yields

$$r = \ln\theta + 1.$$

This also proves the optimality of the *threshold-based* online algorithm, as the competitive ratio for the fractional case is lower bounded by $\ln\theta + 1$.

Assuming x is an real number that is not less than l, the threshold value to further admit packet when the queue length is x, denoted by $g(x)$, is

$$g(x) = \alpha rm \left(\frac{M}{\alpha rm}\right)^{\frac{(x-l)n}{Bn-\alpha Bn}}.$$

Let $n \to \infty$ and replace r with $\ln\theta + 1$, we have

$$g(x) = me^{(\ln\theta+1)\frac{x}{B}-1}.$$

□

A.3 Proof of Lemma 4.8

Proof. Here we only prove the nontrivial case where $b \geq l$. Assume the initial queue length is 0 and at time slot $t^{\max}$, $b_\omega^{\text{gOnAlg}}(t)$ reaches the maximum value during the time period $[1, T]$, i.e., $b_\omega^{\text{gOnAlg}}(t^{\max}) = b$. We consider a special worst instance where there is no packet delivered during $[1, t^{\max}]$ and no packet buffered by gOnAlg during $[t^{\max} + 1, T]$. In this case, we have the following claimed results:

(1) The profit earned by gOnAlg is at least $\sum_{i=1}^{b} v_i$. This is due to the fact that the investigated online algorithm is defined by a non-decreasing threshold values. It is easy to see that the minimum profit earned by gOnAlg is $\sum_{i=1}^{b} v_i$ when the queue length increases from 0 to b.

(2) The profit by OPT is at most $v_{b+1}B + \sum_{i=l+1}^{b} v_i$. As analyzed in the proof for Lemma 4.2, there is no packet with value larger than or equal to v_{b+1} during $[1, T]$. So under the worst case, OPT will definitely buffer B packets with the value of $v_{b+1} - \delta$ where δ is an arbitrarily small positive value. This happens only when $b_{\omega}^{\text{gOnAlg}}(t)$ reaches b. Because we assume there are no packets delivered during $[1, t^{\max}]$, thus OPT will not buffer packet before $t^{\max}$. After OPT buffers B packets with value $v_{b+1} - \delta$, the number of packets further buffered by OPT before t ($t > t^{\max}$) will not be larger than b minus the queue length under gOnAlg, i.e., $b - b_{\omega}^{\text{gOnAlg}}(t)$, since there are no packets buffered by gOnAlg during $[t^{\max} + 1, T]$ as assumed. The packet value will not be larger than the threshold of gOnAlg, so the maximum profit earned by OPT during the time period when $b_{\omega}^{\text{gOnAlg}}(t)$ goes back to 0 from b is not larger than $\sum_{i=l+1}^{b} v_i$. Concluding the above, we can get that the maximum profit by OPT during $[1, T]$ will not be larger than $v_{b+1}B + \sum_{i=l+1}^{b} v_i$.

Based on the above analysis, we know that the maximum profit ratio of OPT and gOnAlg when there is no packet delivered during $[1, t^{\max}]$ and no packet buffered during $[t^{\max} + 1, T]$ is at most $\left(v_{b+1}B + \sum\limits_{i=l+1}^{b} v_i\right) / \left(\sum\limits_{i=1}^{b} v_i\right)$. Actually, this ratio can be approximately realized by the input instance in Eq. (19) (also shown in Figure 5).

$$[(v_1, 0)\times l, (v_2, 0), \dots, (v_b, 0), \underbrace{(v_{b+1}-\delta, 0)\times B, (v_b-\delta, 1), (v_{b-1}-\delta, 1), \dots, (v_{l+1}-\delta, 1)}_{\text{The time slots that OPT selects to buffer.}}]. \tag{19}$$

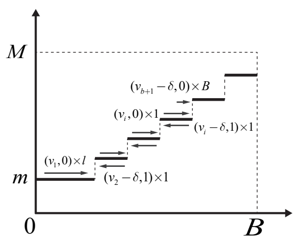

Fig. 5. The constructed worst case.

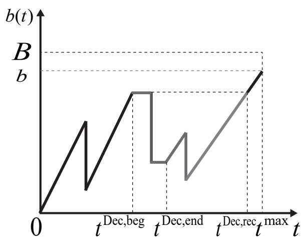

Fig. 6. The delivery time period.

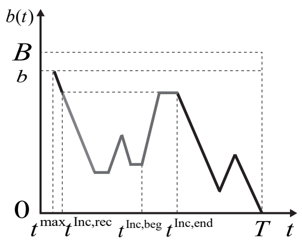

Fig. 7. The buffering time period.

For the case of allowing packet delivery during $[1, t^{\max}]$ or packet buffering during $[t^{\max} + 1, T]$, we have the following analysis:

(1) Assume under the worst case ω, there is a time period $[t^{\text{Dec,beg}}, t^{\text{Dec,end}}] \in [1, t^{\max}]$ when $b_{\omega}^{\text{gOnAlg}}(t)$ keeps decreasing or unchanged due to the packet delivery. Without loss of generality, we assume $[t^{\text{Dec,beg}}, t^{\text{Dec,end}}]$ is the time period of which the beginning queue length $b_{\omega}^{\text{gOnAlg}}(t^{\text{Dec,beg}})$ (for simplicity, we denote $b_{\omega}^{\text{gOnAlg}}(t^{\text{Dec,beg}})$ as b^{Dec}) is the largest among decreasing periods, and assume at time slot $t^{\text{Dec,rec}}$, the queue length returns to b^{Dec}. For details of the above instance, readers can also refer to Figure 6. Because there's no time period with packet delivery after $t^{\text{Dec,rec}}$ and OPT will definitely empty the buffer before the queue length under gOnAlg reaches the maximum, thus $b_{\omega}^{\text{OPT}}(t^{\text{Rec}}) = 0$ must hold. Further, there's no packet of value larger than $g(b^{\text{Dec}})$ before $t^{\text{Dec,beg}}$, so we have $b_{\omega}^{\text{OPT}}(t^{\text{Dec,beg}} - 1) = 0$ for some worst case. From the above analysis, we have that the queue length of gOnAlg and OPT at $t^{\text{Dec,rec}}$ will both return to the state at $t^{\text{Dec,beg}}$. Then under some particular worst case, the profit ratio during $[t^{\text{Dec,beg}}, t^{\text{Dec,rec}}]$ must be larger than $\text{CR}_b(\text{gOnAlg})$ (otherwise, we just "delete" the input segment during $[t^{\text{Dec,beg}}, t^{\text{Dec,rec}}]$, and this will not decrease the profit ratio). Moreover, the adversary can repeat the input instance during $[t^{\text{Dec,beg}}, t^{\text{Dec,rec}}]$

for arbitrary times, since the queue length of OPT and gOnAlg both go back to the state at $t^{\text{Dec,beg}}$. Assume the input instance during $[1, t^{\text{Dec,beg}} - 1]$ is $\boldsymbol{\omega}_1$, and the input instance during $[t^{\text{Dec,beg}}, t^{\text{Dec,rec}}]$ is $\boldsymbol{\omega}_2$, we can construct the input instance $\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2 \times j + [(m), B]$ which lies in subset $\Omega_{b^{\text{Dec}}}^{\text{gOnAlg}}$. When j is large enough, we can get a larger local competitive ratio than $\textsc{cr}_b(\text{gOnAlg})$, i.e.,

$$\textsc{cr}_b(\text{gOnAlg}) < \textsc{cr}_{b^{\text{Dec}}}(\text{gOnAlg}),\ b^{\text{Dec}} < b.$$

In conclusion, the above analysis implies that when $\textsc{cr}_b(\text{gOnAlg}) \geq \textsc{cr}_{b'}(\text{gOnAlg})$ for $b' < b$, there will be no packets delivered during $[1, t^{\max}]$.

(2) Assume under the worst case $\boldsymbol{\omega}$, there is a time period $[t^{\text{Inc,beg}}, t^{\text{Inc,end}}]$ when $b_{\boldsymbol{\omega}}^{\text{gOnAlg}}(t)$ keeps increasing or unchanged due to the packet buffering. Similar to the analysis in (1), we can also find the buffering time period with the largest *ending* queue length as shown in Figure 7. We can also deduce that the buffer at time $t^{\text{Inc,rec}}$ and $t^{\text{Inc,end}}$ must be full, since there will be no packet of value larger than $g(b_{\boldsymbol{\omega}}^{\text{gOnAlg}}(t^{\text{Inc,beg}}))$ or $g(b_{\boldsymbol{\omega}}^{\text{gOnAlg}}(t^{\text{Inc,rec}}))$ afterwards. Thus under the worst case, the profit ratio during $[t^{\text{Inc,rec}}, t^{\text{Inc,end}}]$ is larger than $\textsc{cr}_{\Omega_b^{\text{gOnAlg}}}(\text{gOnAlg})$. Then we can construct a case similar to (1) and prove that there is a subset whose local competitive ratio is larger than that of Ω_b^{gOnAlg}. At last, we can get that no packets buffered during $[t^{\max} + 1, T]$ when $\textsc{cr}_b(\text{gOnAlg}) \geq \textsc{cr}_{b'}(\text{gOnAlg})$ for $b' > b$.

Concluding (1) and (2), we can prove that when $\textsc{cr}_b(\text{gOnAlg}) \geq \textsc{cr}_{b'}(\text{gOnAlg}),\ b' > b$, there is no packet delivered during $[1, t^{\max}]$ and no packet buffered during $[t^{\max} + 1, T]$ under the worst case. In this case, the worst input instance should be as shown in Figure 5, and the corresponding competitive ratio is

$$\textsc{cr}_b(\text{gOnAlg}) = \frac{v_{b+1}B + \sum\limits_{i=l+1}^{b} v_i}{\sum\limits_{i=1}^{b} v_i}.$$

This completes the proof. □

A.4 Proof of Lemma 4.9

Proof. Given the length of the first step is l, optimizing the threshold values of $\textsc{cr}(\text{gOnAlg})$ is equivalent to solving the following problem:

$$\begin{array}{lll} \min & y & \\ \text{s.t.} & y \geq \left(v_{b+1}B + \sum\limits_{i=l+1}^{b} v_i\right) / \left(ml + \sum\limits_{i=l+1}^{b} v_i\right), & b = l, l+1, \ldots, B. \\ \text{var} & y,\ m \leq v_i \leq M, & i = l+1, l+2, \ldots, B. \end{array}$$

The vector of variables is $[y, v_{l+1}, v_{l+2}, \ldots, v_B]$. By partially dualizing on the first set of nonequality constraints, we obtain the following Lagrangian function:

$$L(y, \boldsymbol{v}, \boldsymbol{\mu}) = y + \sum_{i=0}^{B-l} \mu_i \left[\left(v_{l+i+1}B + \sum_{j=l+1}^{l+i} v_j\right) - y\left(ml + \sum_{j=l+1}^{l+i} v_j\right)\right],$$

where $\mu_i,\ i = 0, 1, \cdots, B - l$ are the dual variables associated with the non-equality constraints.

The first-order optimality necessary condition can be expressed as

$$1 - \sum_{i=0}^{B-l} \mu_i \left(ml + \sum_{j=l+1}^{l+i} v_j \right) = 0,$$
$$\mu_i B + (1-y) \sum_{k=i+1}^{B-l} \mu_k = 0, \qquad i = 0, 1, \ldots, B-l-1.$$

Note that $m \leq v_j \leq M$ for $j = l+1, l+2, \ldots, B-1$. Note also that y is equal to $\textsc{cr}$(gOnAlg), so definitely, we have $1 - y < 0$. Combining the above equations, we establish the result that $\mu_i > 0,\ i = 0, 1, \ldots, B-l$. Moreover, according to the complementary slackness condition

$$\mu_i \left[\left(v_{l+i+1} B + \sum_{j=l+1}^{l+i} v_j \right) - y \left(ml + \sum_{j=l+1}^{l+i} v_j \right) \right] = 0,\ i = 0, 1, \ldots, B-l,$$

the following equations hold:

$$\left(v_{l+i+1} B + \sum_{j=l+1}^{l+i} v_j \right) - y \left(ml + \sum_{j=l+1}^{l+i} v_j \right) = 0,\ i = 0, 1, \ldots, B-l.$$

Alternatively, theses equations can be rewritten as

$$\frac{v_{l+i+1} B + \sum_{j=l+1}^{l+i} v_j}{ml + \sum_{j=l+1}^{l+i} v_j} = y,\ i = 0, 1, \ldots, B-l.$$

This completes the proof. □

Received August 2017; revised October 2017; accepted December 2017