An Optimal Algorithm for 2-Bounded Delay Buffer Management with Lookahead

Koji M. Kobayashi(✉)

The University of Tokyo, Tokyo, Japan
kojikoba@mi.u-tokyo.ac.jp

Abstract. The bounded delay buffer management problem, which was proposed by Kesselman et al. (STOC 2001 and SIAM Journal on Computing 33(3), 2004), is an online problem focusing on buffer management of a switch supporting Quality of Service (QoS). The problem definition is as follows: Packets arrive to a buffer over time and each packet is specified by the *release time*, *deadline* and *value*. An algorithm can transmit at most one packet from the buffer at each integer time and can gain its value as the *profit* if transmitting a packet by its deadline after its release time. The objective of this problem is to maximize the gained profit. We say that an instance of the problem is s-bounded if for any packet, an algorithm has at most s chances to transmit it. For any $s \geq 2$, Hajek (CISS 2001) showed that the competitive ratio of any deterministic algorithm is at least $(1+\sqrt{5})/2 \geq 1.618$. Very recently, Veselý et al. (SODA 2019) designed an online algorithm matching the lower bound.

Böhm et al. (ISAAC 2016 and Theoretical Computer Science, 2019) introduced the *lookahead* ability to an online algorithm, that is the algorithm can gain information about future arriving packets, and showed that for $s = 2$, there is an algorithm which achieves the competitive ratio of $(-1+\sqrt{13})/2 \leq 1.303$. Also, they showed that the competitive ratio of any deterministic algorithm is at least $(1+\sqrt{17})/4 \geq 1.280$. In this paper, for the 2-bounded model with lookahead, we design an algorithm with a matching competitive ratio of $(1+\sqrt{17})/4$.

1 Introduction

The online buffer management problem proposed by Aiello et al. [1] formulates the management of buffers to store arriving packets in a network switch with Quality of Service (QoS) support as an online problem. This problem has received much attention among online problems and has been studied for the last fifteen years, which leads to developing various variants of this problem (see comprehensive surveys [17,26]). Kesselman et al. [23] proposed the *bounded delay buffer management* problem as one of the variants, whose definition is as follows: Packets arrive to a buffer over time. A packet p is specified by the *release time* $r(p)$, *value* $v(p)$ and *deadline* $d(p)$. An algorithm is allowed to transfer at most one packet at each integer time. If the algorithm transmits a packet between

D.-Z. Du et al. (Eds.): COCOON 2019, LNCS 11653, pp. 350–362, 2019.
https://doi.org/10.1007/978-3-030-26176-4_29

its release time and deadline, it can gain its value as the *profit*. The objective of this problem is to maximize the gained profit. The performance of an online algorithm for this problem is evaluated using *competitive analysis* [11,27]. If for any problem instance, the profit of an optimal offline algorithm OPT is at most c times that of an online algorithm A, then we say that the competitive ratio of A is at most c. We call a problem instance the *s-bounded instance* (or *s-bounded delay buffer management* problem) in which for any packet p, $d(p)-r(p)+1 \leq s$. For any $s \geq 2$, Hajek [19] showed that the competitive ratio of any deterministic algorithm is at least $(1+\sqrt{5})/2 \geq 1.618$. Very recently, Veselý et al. [28] designed an online algorithm matching the lower bound.

There is much research among online problems to reduce the competitive ratio of an online algorithm for the original problems by adding extra abilities to the algorithm. One of the major methods is called the *lookahead* ability, with which an online algorithm can obtain information about arriving packets in the near future. This ability is introduced to various online problems: the bin packing problem [18], the paging problem [2,12], the list update problem [3], the scheduling problem [25] and so on. Then, Böhm et al. [9,10] introduced the lookahead ability to the bounded delay buffer management problem, that is, they gave an online algorithm for this problem an ability to obtain the information about future arriving packets and analyzed its performance.

Previous Results and Our Results. Böhm et al. [9,10] studied the 2-bounded bounded delay buffer management problem with lookahead. They designed a deterministic algorithm whose competitive ratio is at most $(-1+\sqrt{13})/2 \leq 1.303$. Also, they proved that the competitive ratio of any deterministic algorithm is at least $(1+\sqrt{17})/4 \geq 1.280$.

In this paper, we show an online algorithm matching their lower bound for this problem, that is, its competitive ratio is exactly $(1+\sqrt{17})/4$. Since the original bounded delay buffer management problem has been solved completely by Veselý et al. [28] just recently, the bounded delay buffer management problem with lookahead is one of the most important variants which should be solved among several variants of this problem. Our result will help to develop an optimal algorithm for s-bounded instances.

Related Results. In the full version [10] of the paper [9], Böhm et al. studied lower bounds on the competitive ratios of online algorithms with more generalized lookahead. Specifically, the lookahead ability in [9] at a time t enables an online algorithm to obtain the information about packets p such that $r(p) = t+1$. In [10], for a positive integer ℓ, they considered the case where the ability at a time t enables an online algorithm to obtain the information about packets p such that $r(p) \leq t + \ell$. They showed that a lower bound of any deterministic algorithm is $\frac{1+\sqrt{5+8\ell+4\ell^2}}{2\ell+2}$. Moreover, they proved that for any $\ell \geq 1$, a lower bound of any randomized online algorithm is 1.25.

As mentioned above, for the s-bounded delay model *without* lookahead, Hajek [19] showed that the competitive ratio of any deterministic algorithm is at least $(1+\sqrt{5})/2 \geq 1.618$ in the case of $s \geq 2$. Independently, this bound

was also shown in [4,13,29]. Several deterministic algorithms have been developed [5,9,10,14,16,23] and very recently, Veselý et al. [28] designed an optimal online algorithm. Moreover, in the case where an algorithm must decide which packet to transmit on the basis of the current buffer situation, called the *memoryless* case, some results were shown [5,14,16]. The *agreeable deadline* variant has also been studied. In this variant, the larger the release times of packets are, the larger their deadlines are. Specifically, for any packets p and p', $d(p) \leq d(p')$ if $r(p) < r(p')$. The lower bound of $(1+\sqrt{5})/2$ by Hajek [19] is applicable to this variant. Li et al. [21,24] displayed an optimal algorithm, whose competitive ratio matches the lower bound. The case in which for any packet p, $d(p)-r(p)+1 = s$ has also been studied, called the *s-uniform delay* variant, which is a specialized variant of the agreeable deadline variant. The current best upper bound for this variant is $(1+\sqrt{5})/2$ [21,24]. Also, in the case of $s = 2$, Chrobak et al. [15] designed an optimal online algorithm whose competitive ratio is 1.377 [15].

The research on randomized algorithms for the bounded delay buffer management problem has also been conducted extensively [5,6,13,14,20–22]. In the case in which s is general, the current best upper and lower bounds are $e/(e-1) \leq 1.582$ [5,14,22] and $5/4 = 1.25$ [13], respectively, against an oblivious adversary were shown. Upper and lower bounds of $e/(e-1)$ [6,22] and $4/3 \geq 1.333$ [6], respectively, against an adaptive adversary were shown. For any fixed s, lower bounds are the same with the bounds in the case in which s is general while upper bounds are $1/(1-(1-\frac{1}{s})^s)$ [22] against the both adversaries.

A generalization of the bounded delay buffer management problem has been studied, called the *weighted item collection* problem [7,8,22]. In this problem, an online algorithm does not know the deadline of each packet but knows the relative order of the deadlines of packets. Many other variants of the buffer management problem have been studied extensively (see e.g. [17,26]).

2 Model Description

We formally give the definition of the 2-bounded delay buffer management problem with lookahead, which is addressed in this paper. An *input* of this problem is a sequence of phases. Time begins with zero and a phase occurs at an integer time. Each phase consists of three subphases. The first occurring subphase is the *arrival subphase*. At an arrival subphase, arbitrarily many packets can arrive to a buffer. The buffer has no capacity limit and hence, all arriving packets can always be accepted to the buffer. A packet p is characterized by the *release time*, *deadline* and *value*, denoted by $r(p)$, $d(p)$ and $v(p)$ respectively. Arrival times and deadlines are non-negative integers and values are positive reals. $d(p)-r(p) \leq 1$ holds because we focus on 2-bounded instances. The second subphase is the *transmission subphase*. At a transmission subphase, an algorithm can transmit at most one packet from its buffer if any packet. At the transmission subphase at a time t, the algorithm can obtain the information about packets arriving at time $t+1$ using the lookahead ability. The third subphase is the *expiration subphase*. At an expiration subphase, a packet which has reached its deadline is

discarded from its buffer. That is, at the expiration subphase at a time t, all the packets p in the buffer such that $d(p) = t$ are discarded.

The *profit* of an algorithm is the sum of the values of packets transmitted by the algorithm. The objective of this problem is to maximize the gained profit. Let $V_A(\sigma)$ denote the profit of an algorithm A for an input σ. Let OPT be an optimal offline algorithm. We say that the competitive ratio of an online algorithm ON is at most c if for any input σ, $V_{OPT}(\sigma) \leq V_{ON}(\sigma)c$.

3 Matching Upper Bound

3.1 Notation and Definitions

We give definitions before defining our algorithm CompareWithPartialOPT (CP). For any integer time t and any algorithm A, $B_A(t)$ denotes the set of packets in A's buffer immediately before the arrival subphase at time t. That is, each packet p in the set is not transmitted before t such that $t > r(p)$ and $t \leq d(p)$. Let us define an offline algorithm PO, which stands for a Partial OPT, which stores all the packets in the buffer of CP at a time and woks optimally given a subinput from the time. For integer times $t, t' \geq t$ and $t'' \in \{t', t'+1\}$ and an input σ, let $PO(t, t', t'')$ be an offline algorithm such that the set of packets in $PO(t, t', t'')$'s buffer immediately before the arrival subphase at time t is equal to that of $B_{CP}(t)$'s, and if the subinput of σ during time $[t, t']$ is given to $PO(t, t', t'')$, that is, packets p such that $r(p) \in [t, t']$ arrive to $PO(t, t', t'')$'s buffer during time $[t, t']$, then $PO(t, t', t'')$ is allowed to transmit $t'' - t + 1$ packets only from time t to t'' inclusive, that is, at $t'' - t + 1$ transmission subphases, and chooses the packets whose total profit is maximized. If there exist packets with the same value in $PO(t, t', t'')$'s buffer, $PO(t, t', t'')$ follows a fixed tie breaking rule. Also, $P(t, t', t'')$ denotes the set of $t'' - t + 1$ packets transmitted by $PO(t, t', t'')$ during time $[t, t'']$. Note that for any t and $t' \geq t$, the following relations hold because of the optimality of packets transmitted by $PO(t, t', t'')$ during time $[t, t'']$:

$$P(t, t', t') \subseteq P(t, t'+1, t'+1) \tag{1}$$

$$P(t, t', t') \subseteq P(t, t', t'+1) \tag{2}$$

and

$$P(t+1, t', t') \subseteq P(t, t', t'). \tag{3}$$

We define for any t and $i \geq 0$,

$$\{m_i(t)\} = P(t, t+i, t+i) \backslash P(t, t+i-1, t+i-1)$$

and any $i \geq 1$,

$$\{q_i(t)\} = P(t, t+i, t+i+1) \backslash P(t, t+i, t+i).$$

Also, we define

$$P(t, t-1, t-1) = \varnothing.$$

If $m_i(t)$ ($q_i(t)$) does not exist, that is, the above equality is the empty set, then we assume that a packet whose value is 0 is given. This assumption is used to make the description of CP simpler and does not affect the performance of CP. Moreover, if there exists a packet p such that both $v(p) = v(q_1(t))$ holds and either $r(p) = t$ or $p \in B_{CP}(t)$, that is, CP can transmit p at t, then $q_0(t)$ denotes p. Also, we define $m_{01}(t) \in \arg\max\{m_0(t), m_1(t)\}$. Let $V(t, t', t'')$ denote the total value of packets in $P(t, t', t'')$. That is, $V(t, t', t'') = \sum_{p \in P(t,t',t'')} v(p)$. We describe each value in the algorithm definition for ease of presentation as follows: $m_i = m_i(t)$, $m_{01} = m_{01}(t)$, $q_i = q_i(t)$ and, $R = (1 + \sqrt{17})/4$.

3.2 Idea Behind Algorithm Design

In this section, we explain the idea behind designing our algorithm CP for better understanding. Suppose that CP decides which packet to transmit at a time t. Let us assume that at t, the buffer of CP stores all the packets in the buffer of OPT at t. We guarantee that this assumption holds at a time satisfying some conditions in a lemma of the full version of this paper. Due to page limitations, we omit all of the proofs in this paper. The full version of this paper is available at https://arxiv.org/abs/1807.00121. If this assumption holds, CP is able to *detect* two packets OPT transmits at times t and $t+1$. To detect here means that CP calculates which packets OPT transmits at these times cause the worst situation with respect to the profit ratio. Let V be the maximum total value of two packets which OPT transmits at t and $t+1$. CP chooses packets p and p' at t and $t+1$, respectively, from packets which are revealed to CP at t such that $V \leq R(v(p) + v(p'))$ holds. Note that p' may arrive at $t+1$. Although both CP's and OPT's buffers have the same packets at some time, the optimal choice depends on the instance, which in turn depends on CP's choice and thus CP might make a non-optimal choice in general. and hence, CP does not always transmit the packets as the ones which OPT transmits although they have the same packets at t. If CP could choose packets for each $t = 0, 2, 4, \ldots$ to satisfy the above inequality, we could prove that the competitive ratio of CP is at most R. However, this is impossible. For example, suppose that packets p_0, p_1 and p_2 are given at time 0 such that $d(p_0) = 0$, $d(p_1) = 1$ and $d(p_2) = 2$ and no other packets are given further. Also, suppose that CP transmits p_1 and p_2 at times 0 and 1, respectively and OPT transmits p_0, p_1 and p_2 at times 0, 1 and 2,

respectively. In this case, CP does not transmit any packet at time 2 and thus, we cannot prove the above inequality.

Thus, the length of a time interval which CP uses to evaluate its competitive ratio is not fixed (such as 2 mentioned above) but variable as follows. Let us assume again that at a time t, the buffer of CP stores all the packets in the buffer of OPT at t. Also, suppose that CP decides which packet to transmit at a time $t'(\geq t)$ (the fact that $t' \leq t+2$ holds will be shown later by the definition of CP). By this assumption, CP can detect $t'-t+2$ packets transmitted by OPT during the time $[t, t'+1]$ (in some special cases, CP can detect $t'-t+3$ packets transmitted by OPT during $[t, t'+2]$). Let V' be the maximum total value of packets which OPT transmits during this time interval. CP chooses packets p and p' at t' and $t'+1$, respectively, considering the total value U of packets which CP already transmitted during time $[t, t'-1]$ such that $V' \leq R(U+v(p)+v(p'))$ holds. For example, suppose that packets $q_0(0), m_0(0)$ and $m_1(0)$ are given at time 0 whose values satisfy the execution conditions of Case 1.2.3.1 in CP. If CP transmits $m_0(0)$ and $m_1(0)$ at times 0 and 1, respectively, then CP can detect that OPT transmits $q_0(0), m_0(0)$ and $m_1(0)$ at times 0, 1 and 2, respectively. In this case, $t = t' = 0$ holds, and the above inequality holds by the condition of Case 1.2.3.1.

The sequences of packets which OPT transmits during $[t, t'+1]$ (or $[t, t'+2]$) are classified into three categories according to a packet p' which CP transmits at $t'+1$ (this fact will be proved in some lemmas of the full version of this paper): (a) Packets which are given during $[t, t'+1]$ satisfy some conditions and OPT transmits specific packets whose total value is $V(t, t'+1, t'+1)$. (b) If the deadline of p' is $t'+1$, then the total value of packets which OPT transmits during $[t, t'+1]$ is at most $V(t, t'+1, t'+1)$. (c) If the deadline of p' is $t'+2$, then the total value of packets which OPT transmits during $[t, t'+2]$ is at most $V(t, t'+1, t'+2)$. Please refer to Table 1. 'Case' column shows the names of cases executed by CP at t. 'Type' column shows the categories of packet sequences transmitted by OPT during $[t, t'+1]$ (or $[t, t'+2]$). 't' and '$t+1$' in 'CP' column show packets which CP transmits at t and $t+1$, respectively. Similarly, 't', '$t+1$' and '$t+2$' in 'OPT' column show packets which CP detects at time t that OPT transmits at t, $t+1$ and $t+2$, respectively. 'Value' column shows the total value of the packets detected by CP. For example, packets detected at Case 1.2.3.1 are classified into (c). CP transmits $m_0(t)$ and $m_1(t)$ at times 0 and 1, respectively. CP can detect that OPT transmits $q_0(t)(= q_1(t)), m_0(t)$ and $m_1(t)$ at times 0, 1, and 2, respectively, and the total value of these packets is $V(t, t+1, t+2)$. On the other hand, suppose that packets satisfying the condition of Case 1.2.3.2 are given. In this case, at time t, if CP decides which packet to transmit at $t+1$, then a situation in which the above inequality does not hold can occur whichever packet which arrives at or before $t+1$ CP chooses. Note that if this condition is satisfied, then *this situation occurs not only for CP but also any online algorithm*, which causes the definition of CP lengthy. Hence, CP chooses $q_0(t)$ as a packet for the transmission subphase at t, and decides which packet to transmit for the transmission subphase of time $t+2$ after making sure of

packets at $t+1$ with lookahead. That is, CP executes Step 2 at the transmission subphase at $t+1$ to choose packets which CP transmits at $t+1$ and $t+2$.

Similarly to the case at time t, CP chooses packets at $t+1$ considering the value $U = v(q_0(t))$ of the packet $q_0(t)$ which CP transmitted at t so that the above inequality holds. Please refer to the row of Case 2.2.1 in Table 2, which is described in the same manner as the previous one. Suppose that packets are given satisfying the conditions of Case 2.2.1. If CP transmits $m_{01}(t)$ and $m_2(t)$ at times $t+1$ and $t+2$, respectively, then CP can detect that OPT transmits $m_0(t), m_1(t)$ and $m_2(t)$ at times t, $t+1$ and $t+2$, respectively. These packets are classified into (b) and the above inequality holds because of the condition of Case 2.2.1. Unfortunately, suppose that packets satisfying the condition of Case 2.2.2.3 are given. In this case, at $t+1$, if CP decides which packet to transmit at $t+2$, then a situation in which the above inequality does not hold can also occur. Note that if the conditions of Cases 1.2.3.2 and 2.2.2.3 are satisfied at times t and $t+1$, respectively, then *this situation occurs for any online algorithm*. Thus, CP chooses $m_0(t)$ as a packet for $t+1$, and executes Step 3 at $t+2$ to choose packets which CP transmits at $t+2$ and $t+3$. Fortunately, as Table 3 shows, at time $t+2$, if CP chooses packets to transmit at $t+2$ and $t+3$ appropriately, then the above inequality holds at any of Cases 3.1 - 3.2.3. Moreover, we will prove that the buffer of CP stores all the packets in the buffer of OPT at $t'+2$ in a lemma of the full version (there exists some exception for a packet sequence classified into (c)). Hence, in the next step, we can regard time $t'+2$ as a new base time, which was time t in the above discussion, and evaluate the profit ratio for each time interval recursively. In this way, CP is designed so that at each time interval $[t, t'+1]$ (or $[t, t'+2]$), the corresponding profit ratio is at most R, that is, its competitive ratio is at most R.

Table 1. Packet prediction at Step 1 at time t

CP				OPT			
Case	Type	t	$t+1$	Value	t	$t+1$	$t+2$
1.1	a, b	m_0		$V(t,t,t)$	m_0		
1.2.1	b	m_1	m_0	$V(t,t+1,t+1)$	m_1	m_0	
1.2.2	b	m_0	m_1	$V(t,t+1,t+1)$	m_0	m_1	
1.2.3.1	c	m_0	m_1	$V(t,t+1,t+2)$	q_0	m_0	m_1
1.2.3.2		q_0	(Step 2)				

m_i and q_i denote $m_i(t)$ and $q_i(t)$ for ease of presentation. $q_0 = q_1$ by definition.

Table 2. Packet prediction at Step 2 at time $t+1$

CP					OPT				
Case	Type	t	$t+1$	$t+2$	Value	t	$t+1$	$t+2$	$t+3$
2.1	b	q_0	m_0	m_1	$V(t,t+2,t+2)$	m_0	m_1	m_2	
2.2.1	b		m_{01}	m_2	$V(t,t+2,t+2)$	m_0	m_1	m_2	
2.2.2.1	a		m_{01}		$V(t,t+1,t+1)$	m_0	m_1		
2.2.2.2	c		m_{01}	m_2	$V(t,t+2,t+3)$	q_0	m_0	m_1	m_2
2.2.2.3			m_0	(Step 3)					

m_i, m_{01} and q_i denote $m_i(t), m_{01}(t)$ and $q_i(t)$. $q_0 = q_1$ by definition.

Table 3. Packet prediction at Step 3 at time $t+2$

CP						OPT					
Case	Type	t	$t+1$	$t+2$	$t+3$	Value	t	$t+1$	$t+2$	$t+3$	$t+4$
3.1	b	q_0	m_0	m_1	m_2	$V(t,t+3,t+3)$	m_0	m_1	m_2	m_3	
3.2.1	b			m_2	m_3	$V(t,t+3,t+3)$	m_0	m_1	m_2	m_3	
3.2.2	a			m_2		$V(t,t+2,t+2)$	m_0	m_1	m_2		
3.2.3	c			m_2	m_3	$V(t,t+3,t+4)$	q_0	m_0	m_1	m_2	m_3

m_i, m_{01} and q_i denote $m_i(t), m_{01}(t)$ and $q_i(t)$. $q_0 = q_1$ by definition.

3.3 Algorithm

The executions of CP are divided into *stages*. Each stage consists of a single transmission subphase, two consecutive transmission subphases, three consecutive transmission subphases or four consecutive transmission subphases.

CP uses the internal variable s_t for holding the name of a packet which CP transmits at a time t. $s_{t'} = \texttt{null}$ holds at first for any integer t'. CP uses the constant `tmp1` (`tmp2`) if at time t ($t+1$), CP cannot decide which packet to transmit at $t+1$ ($t+2$) in Case 1.2.3.2 (2.2.2.3). On the other hand, once the name of a packet is set to s_{t+1} at time t, CP certainly transmits the packet at $t+1$. It is applied to s_{t+2} (s_{t+3}) which is set at $t+1$ ($t+2$).

CompareWithPartialOPT (CP)

Initialize: For any integer time t', $s_{t'}$:=`null`.
Suppose that a stage starts at a time t.

Step 1 (the transmission subphase at t):
Execute the following cases (Case 1.1 - 1.2.3.2) and transmit the packet s_t. If s_{t+1} =`null` after this transmission (i.e., Case 1.1 is executed), then finish the stage.

Case 1.1 ($d(m_0) = t$ or q_0 does not exist): $s_t := m_0$.
Case 1.2 ($d(m_0) \neq t$):
Case 1.2.1 ($d(m_1) = t$): $s_t := m_1$ and $s_{t+1} := m_0$.
Case 1.2.2 ($d(m_1) = t+1$): $s_t := m_0$ and $s_{t+1} := m_1$.
Case 1.2.3 ($d(m_1) \neq t+1$):
Case 1.2.3.1 $\frac{V(t,t+1,t+2)}{v(m_0)+v(m_1)} \leq R$**):** $s_t := m_0$ and $s_{t+1} := m_1$.
Case 1.2.3.2 $\frac{V(t,t+1,t+2)}{v(m_0)+v(m_1)} > R$**):** $s_t := q_0$ and s_{t+1} := `tmp1`.

Step 2 (the transmission subphase at $t+1$):
If s_{t+1} =`tmp1`, then execute the following cases (Case 2.1 - 2.2.2.3). Transmit the packet s_{t+1}. If s_{t+2} =`null` after this transmission (i.e., Case 2.2.2.1 is executed), then finish the stage.

Case 2.1 ($\frac{V(t,t+2,t+2)}{v(q_0)+v(m_0)+v(m_1)} \leq R$**):** $s_{t+1} := m_0$ and $s_{t+2} := m_1$.
Case 2.2 ($\frac{V(t,t+2,t+2)}{v(q_0)+v(m_0)+v(m_1)} > R$**):**
Case 2.2.1 ($d(m_2) = t+2$): $s_{t+1} := m_{01}$ and $s_{t+2} := m_2$.
Case 2.2.2 ($d(m_2) \neq t+2$):
Case 2.2.2.1 ($v(q_2) \neq v(q_1)$): $s_{t+1} := m_{01}$.
Case 2.2.2.2 ($v(q_2) = v(q_1)$ and $\frac{V(t,t+2,t+3)}{v(q_0)+v(m_{01})+v(m_2)} \leq R$**):** $s_{t+1} := m_{01}$ and $s_{t+2} := m_2$.
Case 2.2.2.3 ($v(q_2) = v(q_1)$ and $\frac{V(t,t+2,t+3)}{v(q_0)+v(m_{01})+v(m_2)} > R$**):** $s_{t+1} := m_0$ and s_{t+2} := `tmp2`.

Step 3 (the transmission subphase at $t+2$):
If s_{t+2} =`tmp2`, then execute the following cases (Case 3.1 - 3.2.3). Transmit the packet s_{t+2}. If s_{t+3} =`null` after this transmission (i.e., Case 3.2.2 is executed), then finish the stage.

Case 3.1 ($\frac{V(t,t+3,t+3)}{v(q_0)+v(m_0)+v(m_1)+v(m_2)} \leq R$**):** $s_{t+2} := m_1$ and $s_{t+3} := m_2$.
Case 3.2 ($\frac{V(t,t+3,t+3)}{v(q_0)+v(m_0)+v(m_1)+v(m_2)} > R$**):**
Case 3.2.1 ($d(m_3) = t+3$): $s_{t+2} := m_2$ and $s_{t+3} := m_3$.
Case 3.2.2 ($d(m_3) \neq t+3$ and $v(q_3) \neq v(q_1)$): $s_{t+2} := m_2$.
Case 3.2.3 ($d(m_3) \neq t+3$ and $v(q_3) = v(q_1)$): $s_{t+2} := m_2$ and $s_{t+3} := m_3$.

Step 4 (the transmission subphase at $t+3$): Transmit s_{t+3} and finish the stage.

3.4 Overview of the Analysis

For ease of analysis, we assume that if CP does not store any packet in its buffer at the transmission subphase at a time t, no packets arrive at or after time $t+1$ any more, that is, the input is over. Note that CP stores no packets but OPT may store one at t, that is, transmit it then. Since we consider a 2-bounded instance, the buffers of OPT and CP are both empty after the expiration subphase at t. This situation is equal to the one before the first packet arrives at time 0 and by the definition of CP, we regard a time at which the buffers are empty as time 0. Therefore, this assumption does not affect the performance of CP.

Consider a given input σ. Let k denote the number of stages after σ is over. Let τ be the last time at which CP transmits a packet. We partition the time sequence $[0, \tau]$ into k sequences T_i $(i = 1, \ldots, k)$ disjointly such that T_i consists of times at which the executions of the ith stage are done. Specifically, if $T_i = [t_i, t'_i]$, then $t_i \leq t'_i$, $t_1 = 0$, $t'_k = \tau$ and for any $j = 2, \ldots, k$, $t_j = t'_{j-1} + 1$. The size of each T_i depends on times at which CP does the executions of the ith stage, that is, which case CP executes at each time: Suppose that $T_i = [t, t']$, in which t and t' are integer times.

- If Case 1.1 is executed at t, then $t' = t$.
- If Case 1.2.1, 1.2.2 or 1.2.3.1 is executed at t, then $t' = t + 1$.
- If Case 1.2.3.2 is executed at t and Case 2.1, 2.2.1 or 2.2.2.2 at $t + 1$, then $t' = t + 2$.
- If Cases 1.2.3.2 and 2.2.2.1 are executed at t and $t + 1$, respectively, then $t' = t + 1$.
- If Cases 1.2.3.2 and 2.2.2.3 are executed at t and $t + 1$, respectively, and Case 3.1, 3.2.1 or 3.2.3 is executed at $t + 2$, then $t' = t + 3$.
- If Cases 1.2.3.2 and 2.2.2.3 are executed at t and $t + 1$, respectively, and Case 3.2.2 is executed at $t + 2$, then $t' = t + 2$.

For a time t, a packet whose release time is t and deadline is $t + 1$ is called a 2_t*-packet*. If for a time t, CP transmits a 2_t-packet p at t and OPT transmits p at $t + 1$, then we call the time $t + 1$ an *extra time* (*e-time*, for short). On the other hand, for each $i = 1, \ldots, k$, let us define T'_i, which is formally defined later, each of which is a subsequence of the time sequence $[0, \tau']$, in which τ' is the last time at which OPT transmits a packet. They are not always disjoint differently from T_i. To analyze the performance of CP, for each $i \in [1, k]$, we will compare the total value of packets transmitted by CP during the time T_i with that by OPT during the time T'_i. T'_i is defined as follows: For $T_i = [t, t']$ in which t and $t'(\geq t)$ are integer times, we define $T'_i = [t, \hat{t}']$, in which if $t' + 1$ is an e-time, then $\hat{t}' = t' + 1$. Otherwise, $\hat{t}' = t'$. We give the lemma about T'_i.

Lemma 1. *A time in* $[0, \tau']$ *is contained in some* T'_i.

For any i, we define an offline algorithm OPT_i to bound the value of packets transmitted by OPT during time $T'_i = [t, t']$, in which t and $t'(\geq t)$ are integer times. Roughly speaking, if t is not an e-time, then OPT_i transmits the same packet as a packet OPT transmits during T'_i. If t is an e-time, then OPT_i

transmits the same packet as a packet OPT transmits during T'_i except for t. However, OPT_{i-1} transmits the same packet as OPT at t.

First, let us define packets in the buffer of OPT_i for $T_i = [t, t']$. If $t = 1$, $B_{OPT_i}(t) = B_{OPT}(t)$. If $t \geq 2$ and t is not an e-time, then $B_{OPT_i}(t) = B_{OPT}(t)$. If $t \geq 2$ and t is an e-time, then $B_{OPT_i}(t) = B_{OPT}(t) \backslash \{p\}$, in which p is the 2_{t-1}-packet which OPT transmits at t. Then, for $T_i = [t, t']$ and $T'_i = [t, \hat{t}']$, we define OPT_i as follows: The subinput of σ during time T'_i is given to OPT_i, that is, packets p such that $r(p) \in T'_i$ arrive to OPT_i's buffer during time T'_i according to their release times. Then, OPT_i is allowed to transmit $\hat{t}' - t + 1$ packets only from time t to $\hat{t}'$ inclusive, that is, at $\hat{t}' - t + 1$ transmission subphases, and chooses the packets whose total profit is maximized. If there exist packets with the same value in OPT_i's buffer, OPT_i follows the same tie breaking rule as OPT.

We use $PO(t, t', \hat{t}')$ to define CP and can evaluate the profit of CP using the profit of $PO(t, t', \hat{t}')$ during T_i. On the other hand, we bound the profit of OPT using that of OPT_i during T'_i. Then, we evaluate the relations between the profit of $PO(t, t', \hat{t}')$ and that of OPT_i. For any $i \in [1, k]$, let V_i denote the total value of packets transmitted by CP during T_i. By definition, $V_{CP}(\sigma) = \sum_{i=1}^{k} V_i$. On the other hand, Lemma 1 indicates that a packet which OPT transmits is transmitted at a time in some $T'_{i'}$ by either $OPT_{i'}$ or $OPT_{i'-1}$. Also, by the definition of OPT_i, if t is not an e-time, OPT_i transmits a packet at t whose value is at least that transmitted by OPT. If t is an e-time, then OPT_{i-1} transmits a packet at t whose value is at least that transmitted by OPT and OPT_i may also transmit a packet. That is, the total value of packets transmitted by OPT_i over all $i \in [1, k]$ is at least that of OPT. For any $i \in [1, k]$, let V'_i denote the total value of packets transmitted by OPT_i during T'_i. Hence, $V_{OPT}(\sigma) \leq \sum_{i=1}^{k} V'_i$. Since

$$\frac{V_{OPT}(\sigma)}{V_{CP}(\sigma)} \leq \frac{\sum_{i=1}^{k} V'_i}{\sum_{i=1}^{k} V_i} \leq \max_{i \in [1,k]} \left\{ \frac{V'_i}{V_i} \right\},$$

we will prove the following lemma:

Lemma 2. *For any $i \in [1, k]$, $V'_i / V_i \leq (1 + \sqrt{17})/4$.*

Therefore, we have the following theorem:

Theorem 1. *The competitive ratio of CP is at most $(1 + \sqrt{17})/4$.*

Acknowledgments. This work was supported by JSPS KAKENHI Grant Number 19K11819.

References

1. Aiello, W., Mansour, Y., Rajagopolan, S., Rosén, A.: Competitive queue policies for differentiated services. J. Algorithms **55**(2), 113–141 (2005)
2. Albers, S.: On the influence of lookahead in competitive paging algorithms. Algorithmica **18**(3), 283–305 (1997)
3. Albers, S.: A competitive analysis of the list update problem with lookahead. Theoret. Comput. Sci. **197**(1–2), 95–109 (1998)
4. Andelman, N., Mansour, Y., Zhu, A.: Competitive queueing policies for QoS switches. In: Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms, pp. 761–770 (2003)
5. Bartal, Y., et al.: Online competitive algorithms for maximizing weighted throughput of unit jobs. In: Diekert, V., Habib, M. (eds.) STACS 2004. LNCS, vol. 2996, pp. 187–198. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24749-4_17
6. Bienkowski, M., Chrobak, M., Jeż, Ł.: Randomized competitive algorithms for online buffer management in the adaptive adversary model. Theoret. Comput. Sci. **412**(39), 5121–5131 (2011)
7. Bienkowski, M., et al.: Collecting weighted items from a dynamic queue. Algorithmica **65**(1), 60–94 (2013)
8. Bienkowski, M., et al.: A Φ-competitive algorithm for collecting items with increasing weights from a dynamic queue. Theoret. Comput. Sci. **475**, 92–102 (2013)
9. Böhm, M., Chrobak, M., Jeż, Ł., Li, F., Sgall, J., Veselý, P.: Online packet scheduling with bounded delay and lookahead. In: Proceedings of the 27th International Symposium on Algorithms and Computation, pp. 21:1–21:13 (2016)
10. Böhm, M., Chrobak, M., Jeż, Ł., Li, F., Sgall, J., Veselý, P.: Online packet scheduling with bounded delay and lookahead. Theoret. Comput. Sci. **776**, 95–113 (2019)
11. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
12. Breslauer, D.: On competitive on-line paging with lookahead. Theoret. Comput. Sci. **209**(1–2), 365–375 (1998)
13. Chin, F.Y.L., Fung, S.P.Y.: Online scheduling for partial job values: does timesharing or randomization help? Algorithmica **37**, 149–164 (2003)
14. Chin, F.Y.L., Chrobak, M., Fung, S.P.Y., Jawor, W., Sgall, J., Tichý, T.: Online competitive algorithms for maximizing weighted throughput of unit jobs. J. Discrete Algorithms **4**(2), 255–276 (2006)
15. Chrobak, M., Jawor, W., Sgall, J., Tichý, T.: Improved online algorithms for buffer management in QoS switches. ACM Trans. Algorithms **3**(4), 50:1–50:19 (2007)
16. Englert, M., Westermann, M.: Considering suppressed packets improves buffer management in quality of service switches. SIAM J. Comput. **41**(5), 1166–1192 (2012)
17. Goldwasser, M.: A survey of buffer management policies for packet switches. ACM SIGACT News **41**(1), 100–128 (2010)
18. Grove, E.F.: Online bin packing with lookahead. In: Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms, pp. 430–436 (1995)
19. Hajek, B.: On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. In: Proceedings of the 35th Conference on Information Sciences and Systems, pp. 434–438 (2001)
20. Jeż, Ł.: Randomized algorithm for agreeable deadlines packet scheduling. In: Proceedings of the 27th Symposium on Theoretical Aspects of Computer Science, pp. 489–500 (2010)

21. Jeż, Ł., Li, F., Sethuraman, J., Stein, C.: Online scheduling of packets with agreeable deadlines. ACM Trans. Algorithms **9**(1), 5:1–5:11 (2012)
22. Jeż, Ł.: A universal randomized packet scheduling algorithm. Algorithmica **67**(4), 498–515 (2013)
23. Kesselman, A., Lotker, Z., Mansour, Y., Patt-Shamir, B., Schieber, B., Sviridenko, M.: Buffer overflow management in QoS switches. SIAM J. Comput. **33**(3), 563–583 (2004)
24. Li, F., Sethuraman, J., Stein, C.: An optimal online algorithm for packet scheduling with agreeable deadlines. In: Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms, pp. 801–802 (2005)
25. Motwani, R., Saraswat, V., Torng, E.: Online scheduling with lookahead: multipass assembly lines. INFORMS J. Comput. **10**(3), 331–340 (1998)
26. Nikolenko, S.I., Kogan, K.: Single and Multiple Buffer Processing. In: Kao, M.Y. (ed.) Encyclopedia of Algorithms. Springer, New York (2016). https://doi.org/10.1007/978-1-4939-2864-4_535
27. Sleator, D., Tarjan, R.: Amortized efficiency of list update and paging rules. Commun. ACM **28**(2), 202–208 (1985)
28. Veselý, P., Chrobak, M., Jeż, Ł., Sgall, J.: A ϕ-competitive algorithm for scheduling packets with deadlines. In: Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms, pp. 202–208 (2019)
29. Zhu, A.: Analysis of queueing policies in QoS switches. J. Algorithms **53**, 123–142 (2004)