Competitive Buffer Management for Multi-Queue Switches in QoS Networks Using Packet Buffering Algorithms

Koji M. Kobayashi[1], Shuichi Miyazaki[2] and Yasuo Okabe[2]
[1]National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan
Corresponding author: kobaya@nii.ac.jp
[2]Academic Center for Computing and Media Studies, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, Japan

Abstract

In this paper, we consider the online buffer management problem, which formulates the problem of managing network switches supporting Quality of Service guarantee. We improve competitive ratios of the 2-value multi-queue switch model, where the value of a packet is restricted to 1 or $\alpha(\geq 1)$. We use a similar approach as Azar and Richter (STOC 2003 and Algorithmica 43(1-2), 2005) did for the multi-value multi-queue switch model. Namely, we show that the competitive ratio of "the relaxed model" of the 2-value multi-queue switch model is at most $x = \min\{c + \frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}$, if the competitive ratio of an online algorithm for the unit-value multi-queue switch model is at most c. Azar and Richter's technique implies that if the competitive ratio of the 2-value single-queue switch model is x', then the competitive ratio of the 2-value multi-queue switch model is at most xx'. We obtain several results using known c and x'.

Keywords: online buffer management, competitive analysis, multi-queue switch, scheduling algorithm

1 Introduction

A great amount of work has been done in order to guarantee Quality of Service (QoS) on the Internet. One possible way of supporting QoS is differentiated services (DiffServ), on which a traffic descriptor assigns a value to each packet according to the importance of the packet. QoS switches then try to decide acceptance/rejection and/or the order of packet transmission using priority values. The goal of the buffer management algorithm is to maximize the total value of transmitted packets.

Recently, this kind of problem has been modeled as an online problem. Many models have been proposed and the most basic one is the *single-queue model* defined as follows [1]: A switch has a FIFO buffer of bounded size B. An input is a sequence of events. Each event is an arrival event or a send event. At an arrival event, one packet arrives at an input port. Each packet has the priority value and the size (the size is always one in this simplest case). A switch can store packets

A preliminary version of this paper was presented at the 21st ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2009.

provided that the total size of stored packets does not exceed B, that is, a switch can store up to B packets simultaneously. At an arrival event, if the buffer is full, the new packet is rejected. If there is a room for the new packet, an online policy determines, without knowledge of the future, whether to accept it or not. (There is a more general *preemptive* model, introduced by Kesselman et al. [23], which allows an algorithm to preempt packets (i.e., to drop packets already in the buffer to make space).) At each send event, the packet at the head of the queue is transmitted. The goal of the problem is to maximize the sum of the values of transmitted packets. A goodness of an online policy is evaluated using *competitive analysis* [14, 38]. If, for any input σ, an online policy ALG obtains at least $1/c$-fraction of the value of an optimal offline policy for σ, then we say that ALG is c-competitive.

Up to the present, several models have been considered. Among them, Azar and Richter [9] have introduced the *multi-queue switch model*. In this model, a switch consists of m input ports and one output port, and each packet has a destination port. Each port has a FIFO queue, which can simultaneously store up to B packets. An input is a sequence of events. Each event is an arrival event or a scheduling event (which is similar to the send event described above). When a packet arrives at an arrival event, an online policy determines to accept it (if the buffer has room for the new packet), reject it, or preempt a packet and accept the new packet. (We consider both models in which preemption is allowed and not.) At a scheduling event, an online policy chooses one nonempty buffer and transmits the first packet of the queue through the output port.

Previous Results. Several results on the competitiveness of the multi-queue switch model have been presented [4, 8, 9, 10, 36]. Table 1 summarizes the current best upper and lower bound results (together with our results) for the 2-value multi-queue switch models, where a packet can take one of two values 1 and $\alpha \geq 1$.

In [9], the authors proposed a technique to convert an online algorithm for the single queue model into one for the multi-queue switch model, so that the competitive ratio of the latter is at most twice that of the former. More formally, they defined a non-FIFO variant, called the *relaxed model*, of the multi-queue switch model (which will be formally defined in Sec. 2.2). They showed that if (i) the competitive ratio of the single queue model is at most x and (ii) the competitive ratio of the preemptive relaxed model is at most x', then the competitive ratio of the corresponding multi-queue switch model is at most xx'. They proved that the competitive ratio of a greedy algorithm for the relaxed model is at most 2, and combining this with the results for the single-queue models (Table 2), they obtained upper bounds described in Table 1.

Our Results. In this paper, we focus on the *2-value* multi-queue switch model and present two algorithms for the preemptive relaxed model. One is Dual Scheduling (DS) and the other is Simple Scheduling (SS). SS uses a deterministic online algorithm A for the unit-value multi-queue switch model as a subroutine, namely SS regards all the arriving packets as 1-packets, and performs exactly the same operation as A does. It is not hard to see that if A is c-competitive then SS is αc-competitive, because the number of packets transmitted by SS is at least $1/c$-fraction of those transmitted by an optimal offline algorithm. DS, which also uses A as a subroutine, is rather complicated (its description is given in Sec. 3.1). We show that if A is c-competitive, then DS is $(c + \frac{2-c}{\alpha(2-c)+c-1})$-competitive. Therefore, we obtain an online algorithm for the relaxed model of the 2-value multi-queue switch model whose competitive ratio is at most $\min\{c + \frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}$. Using the result of Azar and Richter [9], we can conclude that there is an online algorithm for the 2-value multi-queue switch model whose competitive ratio is at most $\min\{c + \frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}c'$

Table 1: Competitive ratios for the 2-value multi-queue switch models

	Non-Preemptive		Preemptive	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
Deterministic Algorithm	$1 + \frac{1}{\alpha \ln(\alpha/(\alpha-1))}$ [20]	$4 - \frac{2}{\alpha}$ * [9] **3.177* (C5.1)**	$\frac{e}{e-1} \approx 1.581$ [4]	2.564*, 2.6† [9] **2.516* (C5.3)** **2.5996§ (C5.2)**
Randomized Algorithm	$\frac{e}{e-1} \approx 1.581$ [13]		$\frac{e}{e-1} \approx 1.581$ [13]	2.5* [9] **2.270* (C5.4)**

* large enough B, †any B, §$B \geq 3$ Bold values indicate our results, where (C5.x) after each value shows the corresponding Corollary number, and α is chosen to maximize each value.

Table 2: Competitive ratios for the single-queue models

		Non-Preemptive		Preemptive	
		Lower Bound	Upper Bound	Lower Bound	Upper Bound
Deterministic Algorithm	2-value	$2 - 1/\alpha$ [1]	$2 - 1/\alpha$ * [7, 40]	1.281 [23, 39]	1.282* [16]
	multi-value	$\ln(\alpha) + 1$ [7]	$\ln(\alpha) + 2$ ‡ [6]	1.419 [25]	1.733 [16]
Randomized Algorithm	2-value			1.197 [5]	$1 + \alpha^{-\frac{1}{2}} - \alpha^{-1}$ * [5]
	multi-value	$\ln(\alpha)/2 + 1$ [40]		1.197 [5]	$7/4 = 1.75$ * [5]

* large enough B, ‡ $B \geq \ln(\alpha) + 2$

if there is a c'-competitive online algorithm for the 2-value single-queue switch model. Using the currently best values for c and c', we obtain several improved upper bounds listed in Sec. 5. Some major results are summarized in Table 1.

Note that Azar and Richter [9] showed that improving competitive ratios for the single-queue models implies improving competitive ratios for the multi-queue switch model. Our results in this paper give additional potential: Improving competitive ratios for the unit-value multi-queue switch models also implies improving competitive ratios for the 2-value multi-queue switch models.

Related Results. There is also work focusing on the multi-value multi-queue switch model [9, 10, 21]. In this model, $\alpha(\geq 1)$ is the ratio between the largest and the smallest values of packets. The current best lower and upper bounds for the non-preemptive case are $\ln(\alpha) + 1$ [7] and $2\ln(\alpha) + 4$ [9], respectively, and those for the preemptive case are $\frac{e}{e-1} \approx 1.581$ [4] and $3 - 1/\alpha$ [21], respectively. Al-Bawani and Souza [2] and Kawahara et al. [22] studied the m-value model where arriving packets have one of m fixed values $\alpha_1, \ldots, \alpha_m$, and packets with value α_i arrive at the ith queue.

For the unit-value multi-queue switch model, Azar and Richter [9] gave a lower bound $1.366 -$

$\Theta(1/m)$ of randomized algorithms for any B and m, and an upper bound $\frac{e}{e-1} + o(1)(\approx 1.582)$ of a randomized algorithm for any B and m. Albers and Schmidt [4] and Schmidt [37] showed that no greedy algorithm can be better than $(2 - 1/B - \Theta(m^{-1/(2B-2)}))$-competitive for any B and large enough m. They also gave a $17/9(\approx 1.889)$-competitive deterministic algorithm for large enough B. Moreover, they showed that for $B = 2$, the same algorithm achieves the competitive ratio of $13/7(\approx 1.858)$ and gave a matching lower bound for large enough m. Schmidt [37] also proposed a deterministic algorithm whose competitive ratio is at most $17/9$ for any even $B \geq 4$ and at most $17/9 + 2/9(B + 1)$ for any odd $B \geq 3$. Furthermore, he showed a lower bound $\frac{e}{e-1}(\approx 1.581)$ of deterministic online algorithms for any B and large enough m, and a lower bound 1.465 of randomized online algorithms for any B and large enough m. Azar and Litichevskey [8] showed a $\frac{e}{e-1}(\approx 1.58)$-competitive deterministic algorithm for large enough $B > \log m$. Schmidt [36] claimed he showed a 3/2-competitive randomized algorithm, whose flaw was pointed out in [13]. Also, in the case of $m = 2$, Schmidt [36] showed a lower bound of $16/13 \approx 1.230$ for any online algorithm for large enough B. Bienkowski and Madry [12] and Kobayashi et al. [32] proved 16/13-competitive algorithms for the randomized and deterministic cases respectively. Bienkowski [13] showed a lower bound of $\frac{e}{e-1}$ for any online algorithm for any B and large enough m.

As for the single-queue models, the current upper and lower bounds on competitive ratios are summarized in Table 2. There are several other models, such as shared-memory switches [19, 24, 31], CIOQ switches [26, 11, 27, 30, 3] and crossbar switches [28, 29, 3], are also extensively studied. Some of the recent models focus on generalizing processing times, in which each packet has its own processing time and it needs this amount of time for transmission [17, 33, 15]. There are comprehensive surveys on the buffer management problems (see e.g. [18, 35]).

2 Preliminaries

In this section, we formally define the online buffer management problem for the 2-value multi-queue switch model and the 2-value relaxed model introduced in [9].

2.1 2-Value Multi-Queue Switch Model

A multi-queue switch has m input ports (FIFO queues), each of which is equipped with a buffer of size B. The size of a packet is one, and hence each port can store up to B packets simultaneously. Each packet has its *value* corresponding to the priority. In the 2-value multi-queue switch model, each packet takes one of two values, say, 1 and $\alpha(\geq 1)$. We assume that the value of α is known to an algorithm in advance. We call a packet with value 1 (α respectively) a 1-*packet* (an α-*packet* respectively). (In the unit-value multi-queue switch model, the value of any packet is identical, say 1.)

An *input* is a sequence of *events*. An *event* is an *arrival event* or a *scheduling event*. We assume that for any given input, no more than one event occurs simultaneously. At an arrival event, a packet (say, p) arrives at one of m input ports, and the task of an algorithm (or a policy) is to choose one of the following actions: insert p into the corresponding queue (*accept* p), drop p (*reject* p), or drop a packet p' existing in the current buffer (*preempt* p') and accept p. Note that we consider in this paper both preemptive and non-preemptive cases. If a packet is accepted, it is stored at the tail of the corresponding input queue. At a scheduling event, an algorithm chooses one nonempty input port from m ones and transmits the packet at the head of the chosen queue. We assume that sufficiently many scheduling events occur to transmit all arriving packets.

The *gain* of an algorithm is the sum of the values of transmitted packets, and our goal is to maximize it. The gain of an algorithm ALG for an input σ is denoted by $V_{ALG}(\sigma)$. For an online algorithm ALG, if for any input σ, $V_{ALG}(\sigma) \geq V_{OPT}(\sigma)/c$, then we say that ALG is *c-competitive*, where OPT is an optimal offline algorithm for σ. Without loss of generality, we can assume that an optimal offline algorithm greedily accepts arriving packets and preempts a packet only when its buffer is full.

2.2 Preemptive 2-Value Relaxed Model

The preemptive 2-value relaxed model is the same as the usual preemptive 2-value multi-queue switch model defined in Sec. 2.1, except for the following relaxation: In the original model, only a packet at the *head* of a queue can be transmitted at a scheduling event, but in the relaxed model, any packet can be transmitted (that is, the buffer is not a FIFO queue).

As is the case with the preemptive multi-queue switch model, we can assume without loss of generality that an optimal offline algorithm greedily accepts arriving packets and preempts a packet only when its buffer is full.

2.3 Notation and Definitions

We give some notation and definitions used throughout this paper. For simplicity, the 2-value multi-queue switch model (the unit-value multi-queue switch model and the preemptive 2-value relaxed model, respectively) is denoted by M_2 (M_1 and M_r respectively). Note that M_2 can be used to describe both the preemptive and non-preemptive models. In addition, let us write OPT_1 and OPT_r as optimal offline algorithms for M_1 and M_r respectively. Let an *event time* denote a time at which at least one event occurs. In particular, we call it an *arrival (scheduling) event time* if the event is an arrival (scheduling) event. Also, let a *non-event time* denote a time at which no event occurs. For an event time t, t_- represents a non-event time between t and the previous event time. Similarly, t_+ is a non-event time between t and the next event time. These definitions are introduced to specify how many packets exist in a buffer immediately before and after an event. The jth queue of the switch is denoted as $Q^{(j)} (1 \leq j \leq m)$. For an algorithm ALG for M_r or M_1, $h^{(j)}_{ALG}(t)$ denotes the number of packets ALG holds in $Q^{(j)}$ at a non-event time t.

3 Algorithm DS

3.1 Dual Scheduling Algorithm

Recall that our target is a non-FIFO model M_r. The full description of our algorithm DS is given in Fig. 3.1, but we first sketch an outline of its behavior. DS uses an online algorithm A (whose competitive ratio is at most c) for M_1 as a subroutine. Without loss of generality, we assume that A is *work-conserving* [9], that is, A transmits a packet at a scheduling event whenever its buffer is nonempty. We also assume that A greedily accepts arriving packets and can preempt a packet only when its buffer is full. It is easy to convert any online algorithm as such without degrading the competitive ratio.

DS uses A in two different ways, one for scheduling of α-packets and the other for 1-packets. Therefore, it is convenient for us to think that DS is equipped with two As. To distinguish these two algorithms, we call them in different names, "AS" for scheduling of α-packets and "OS" for scheduling of 1-packets.

At an arrival event (Case A), DS accepts packets greedily, by prioritizing α-packets over 1-packets. At the same time, it runs AS and OS as follows. If an arriving packet is an α-packet, then DS makes two copies of it and passes each copy to AS and OS. If an arriving packet is a 1-packet, then DS makes one copy and passes it to OS only. (From now on, we do not distinguish between copies and original.) Both AS and OS then accept the passed packets greedily.

One additional operation is needed. If DS accepts a 1-packet p but OS rejects p (because its buffer is full), then DS chooses a packet, say p', from OS's current buffer and associates it with p. This correspondence is written as $X(p') = p$ using a function X. This operation can be interpreted as follows: For a treatment of a 1-packet p at a scheduling event, DS refers to the behavior of OS for the same packet p. But since OS rejected p, DS needs an alternative packet to refer to, and p' is selected for this purpose.

At a scheduling event (Case S), DS refers to the behavior of either AS or OS, depending on the contents of its buffer. If DS has at least one α-packet (Case AS), it runs AS for scheduling events until either AS transmits an α-packet that DS has, in which case DS transmits the same packet, or AS's buffer gets empty, in which case DS does not transmit a packet. If DS has no α-packet (Case OS), it essentially does the same operation using OS by taking the function X into consideration. Namely, it runs OS for scheduling events until either OS transmits a packet p such that DS has $X(p)$, in which case DS transmits $X(p)$, or OS's buffer gets empty, in which case DS does not transmit a packet.

3.2 Feasibility of DS

In this section, we prove the feasibility of Case A2.3.

Lemma 3.1 *Let t be an arrival event time such that (i) a 1-packet p arrives at $Q^{(\ell)}$ at t, (ii) OS stores B packets at $Q^{(\ell)}$ at t_-, and (iii) DS accepts p. Then, there exists at least one packet p' such that OS holds p' but DS does not hold $X(p')$ in $Q^{(\ell)}$ at t_-.*

Proof. Since DS accepts p at t by the condition (iii), $h^{(\ell)}_{DS}(t_-) < B$ by the definition of buffer management of DS. $h^{(\ell)}_{OS}(t_-) = B$ by the condition (ii). Hence, there must be at least one packet p' such that OS stores p' but DS does not store $X(p')$. □

3.3 Basic Properties of DS

In this section, we show several lemmas that relate behaviors of DS and its subroutines AS and OS. Among them, Lemma 3.4 is important because it relates the number of packets transmitted by these algorithms.

Recall that in Case OS1.1, DS transmits the packet $X(p)$ when OS transmits a packet p. In this case, we say that "*OS returns p to DS*".

Lemma 3.2 *For a non-event time τ on σ_r, if DS stores a 1-packet p at τ, then OS stores p' such that $X(p') = p$ at τ.*

Proof. We prove the lemma by induction on time. At a time just before the first event, the statement is true because DS does not hold a packet. Let t be an event time on σ_r. We assume that the statement is true at time t_- and show that it is true at t_+.

Dual Scheduling Algorithm

Suppose that an event occurs at a time t.

Case A (t is an arrival event time): (suppose that a packet p arrives at $Q^{(\ell)}$)

***DS*'s execution:** $X(p) := p$. If $h^{(\ell)}_{DS}(t_-) < B$, DS accepts p. If p is an α-packet, $h^{(\ell)}_{DS}(t_-) = B$ and DS stores a 1-packet in $Q^{(\ell)}$ at t_-, DS preempts it and accepts p. Otherwise, DS rejects p. If p is an α-packet, DS passes two copies of p to AS and OS. Otherwise, DS passes a copy of p to OS.

***AS*'s execution:** If AS's buffer is not full, AS accepts p. Otherwise, AS rejects p.

***OS*'s execution:** If OS's buffer is not full, OS accepts p. Otherwise, OS rejects p.

***DS*'s execution:** If $h^{(\ell)}_{OS}(t_-) = B$ and DS accepts p, DS sets $X(p') := p$ in which p' is a packet such that OS stores a packet p' in $Q^{(\ell)}$ at t_- but DS does not store $X(p')$ in $Q^{(\ell)}$ at t_-. Such p' exists by Lemma 3.1.

Case S (t is a scheduling event time):

Case AS (DS stores at least one α-packet): Let AS execute a scheduling event. If AS transmits a packet p, then go to Step AS1. Otherwise (i.e. if AS does not transmit a packet), do nothing and finish.

Step AS1: Execute one of the following two cases.

Case AS1.1 (DS stores p at t_-): DS transmits p and finish.

Case AS1.2 (DS does not store p at t_-): Go back to Case AS.

Case OS (DS does not have any α-packet): Let OS execute a scheduling event. If OS transmits a packet p, then go to Step OS1. Otherwise (i.e. if OS does not transmit a packet), do nothing and finish.

Step OS1: Execute one of the following two cases.

Case OS1.1 (DS stores $X(p)$ at t_-): DS transmits $X(p)$ and finish.

Case OS1.2 (DS does not store $X(p)$ at t_-): Go back to Case OS.

Figure 1: Dual Scheduling Algorithm

Case 1: t is an arrival event time: Let p be a 1-packet which arrives at $Q^{(i)}$ at t. If DS rejects p at t, the statement is true. Hence, suppose that DS accepts p. If $h^{(i)}_{OS}(t_-) < B$, OS accepts p because of its greediness and $X(p) = p$ holds by definition. If $h^{(i)}_{OS}(t_-) = B$, DS sets $X(p') := p$ for some packet p' in OS's buffer, which can be executed by Lemma 3.1.

Case 2: t is a scheduling event time: If either DS executes Case AS at t or OS does not transmit a packet p'' at t such that DS stores $X(p'')$ at t_-, then the statement is true by the induction hypothesis. If OS transmits a 1-packet p'' such that DS stores $X(p'')$ at t_- when DS executes Case OS at t, then OS returns p'' to DS in Case OS1.1 and DS transmits $X(p'')$ at t.

We have shown that the statement is true at time t_+. □

Let σ_r be an input for DS. Let $f_{AS}(\sigma_r)$ be an input constructed by removing all the arrival events of 1-packets from σ_r. Next, for each scheduling event e of $f_{AS}(\sigma_r)$, if DS executes Case AS1.2 k times at e, then insert k new scheduling events just after e. Let $g_{AS}(\sigma_r)$ be the resulting input. As we have mentioned in Sec. 3.1, we assume that the subroutine A's acceptance policy is greedy. Also, recall that only α-packets are passed to AS, and AS accepts them greedily and may execute more than one scheduling events at one scheduling event of σ_r. It is then not hard to see that AS's behavior on σ_r as a subroutine of DS is identical to the behavior of AS on

$g_{AS}(\sigma_r)$. (Later it will be shown that DS never executes Case AS1.2 and hence it appears to be $g_{AS}(\sigma_r) = f_{AS}(\sigma_r)$.) We also define $f_{OS}(\sigma_r)$ and $g_{OS}(\sigma_r)$. Let $f_{OS}(\sigma_r)$ be the input constructed from σ_r by removing all the scheduling events where DS executes Case AS. The construction of $g_{OS}(\sigma_r)$ from $f_{OS}(\sigma_r)$ is exactly the same as construction of $g_{AS}(\sigma_r)$ from $f_{AS}(\sigma_r)$. For example, suppose that $f_{OS}(\sigma_r) = (e_1, e_2, e_3, e_4, e_5, e_6)$, where e_2, e_4 and e_6 are scheduling events and the others are arrival events, and suppose that DS executes Case OS1.2 twice, never and once when dealing with e_2, e_4 and e_6 respectively. In this case, $g_{OS}(\sigma_r) = (e_1, e_2, e'_2, e''_2, e_3, e_4, e_5, e_6, e'_6)$, where e'_2, e''_2 and e'_6 are additional scheduling events. We remark that introducing $g_{AS}(\sigma_r)$ and $g_{OS}(\sigma_r)$ allows us to analyze behaviors of AS and OS independently, and hence makes the analysis of the competitive ratio of DS easier. Although we have mentioned in Sec. 2.3 that no two events occur simultaneously, in $g_{OS}(\sigma_r)$ two or more scheduling events may occur simultaneously. This is only for the convenience of analysis.

Lemma 3.3 *For a non-event time τ on σ_r, $\forall i\ h^{(i)}_{OS}(\tau) \geq h^{(i)}_{DS}(\tau)$.*

Proof. We prove the lemma by induction on time. At a time just before the first event, the statement is true because DS does not store a packet. Let t be an event time on σ_r. We assume that the statement is true at time t_- and show that it is true at t_+, namely, we assume that $\forall i\ h^{(i)}_{OS}(t_-) \geq h^{(i)}_{DS}(t_-)$ and show that $\forall i\ h^{(i)}_{OS}(t_+) \geq h^{(i)}_{DS}(t_+)$.

Case 1: t is an arrival event time: Let p be a packet which arrives at $Q^{(j)}$ at t. By the induction hypothesis, clearly $\forall i(\neq j)\ h^{(i)}_{OS}(t_+) \geq h^{(i)}_{DS}(t_+)$. Thus, in what follows, we will show $h^{(j)}_{OS}(t_+) \geq h^{(j)}_{DS}(t_+)$.

If DS rejects p at t, we have $h^{(j)}_{DS}(t_+) = h^{(j)}_{DS}(t_-)$ and $h^{(j)}_{OS}(t_+) \geq h^{(j)}_{OS}(t_-)$. Since $h^{(j)}_{OS}(t_-) \geq h^{(j)}_{DS}(t_-)$ by the induction hypothesis, the statement is true. If DS accepts p at t, $h^{(j)}_{DS}(t_+) = h^{(j)}_{DS}(t_-) + 1$. If $h^{(j)}_{OS}(t_-) < B$, OS accepts p, that is, $h^{(j)}_{OS}(t_+) = h^{(j)}_{OS}(t_-) + 1$. Hence, the statement is true by the induction hypothesis. If $h^{(j)}_{OS}(t_-) = B$, then $h^{(j)}_{OS}(t_+) = h^{(j)}_{OS}(t_-) = B$. Clearly $h^{(j)}_{OS}(t_+) = B \geq h^{(j)}_{DS}(t_+)$.

Case 2: t is a scheduling event time and DS executes Case AS: By definition, the number of arriving α-packets in $f_{AS}(\sigma_r)$ is equal to that in σ_r, and moreover both DS and AS greedily accept α-packets. Hence, a packet AS transmits at Case AS is stored in DS's buffer at the same time, which means that DS does not execute Case AS1.2. Thus, in this case, DS transmits exactly one α-packet at t. Hence, $h^{(j)}_{DS}(t_+) = h^{(j)}_{DS}(t_-) - 1$, where DS transmits the α-packet from $Q^{(j)}$. On the other hand, since there does not exist a scheduling event at t for OS by the definition of $f_{OS}(\sigma_r)$, OS does not transmit a packet at t. Hence, $\forall i\ h^{(i)}_{OS}(t_+) = h^{(i)}_{OS}(t_-)$ holds. Therefore, by the induction hypothesis, the statement holds.

Case 3: t is a scheduling event time and DS executes Case OS: Note that more than one scheduling event may occur at one time on input $g_{OS}(\sigma_r)$, and OS may transmit several packets at that time. Let x_i be the number of packets p' such that OS transmits p' from $Q^{(i)}$ at t but DS does not store $X(p')$ in $Q^{(i)}$ at t_-. Also, if OS has at least one 1-packet p'' such that DS stores $X(p'')$ at t_-, a 1-packet is certainly returned to DS at t by the definition of Case OS1.1. (In the following, we assume that this packet is returned from $Q^{(j)}$.) Otherwise, no packet is returned. Let y_i be the number of a returned 1-packet in $Q^{(i)}$ at t. That is, if $i = j$, then $y_i \in \{0, 1\}$. Otherwise, $y_i = 0$. Then,

$$\forall i\ h^{(i)}_{OS}(t_+) = h^{(i)}_{OS}(t_-) - x_i - y_i.$$

All the packets which DS stores at t_- are 1-packets because DS does not store α-packets at t_- by the condition of Case OS. Also, these packets are stored by OS in its buffer at t_- by Lemma 3.2.

Hence,

$$\forall i\ h_{OS}^{(i)}(t_-) \geq h_{DS}^{(i)}(t_-) + x_i.$$

On the other hand, since DS transmits a packet when OS returns one in Case OS,

$$\forall i\ h_{DS}^{(i)}(t_+) = h_{DS}^{(i)}(t_-) - y_i.$$

Therefore by the above argument,

$$\forall i\ h_{OS}^{(i)}(t_+) = h_{OS}^{(i)}(t_-) - x_i - y_i \geq h_{DS}^{(i)}(t_-) + x_i - x_i - y_i = h_{DS}^{(i)}(t_+) + y_i - y_i = h_{DS}^{(i)}(t_+).$$

For each case, we have shown that the statement holds at t_+. □

Let $R_{AS}(\sigma_r)$ ($R_{OS}(\sigma_r)$) be the number of packets which DS transmits when executing Case AS1.1 (Case OS1.1). For a model $M \in \{M_1, M_r\}$, an input σ on M and an algorithm ALG for M, define $T_{ALG}(\sigma)$ as the number of packets transmitted by ALG on σ.

Lemma 3.4 $R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \geq T_{OS}(g_{OS}(\sigma_r))$.

Proof. For a non-event time τ on σ_r, let $R_{AS}(\sigma_r, \tau)$ ($R_{OS}(\sigma_r, \tau)$) denote the number of packets which DS transmits when executing Case AS1.1 (Case OS1.1) for σ_r before τ, and let $T_{OS}(g_{OS}(\sigma_r), \tau)$ denote the number of packets transmitted by OS before τ. To prove this lemma, it suffices to prove the following inequality for any non-event time τ on σ_r:

$$R_{AS}(\sigma_r, \tau) + R_{OS}(\sigma_r, \tau) + \sum_{i=1}^{m} h_{DS}^{(i)}(\tau) \geq T_{OS}(g_{OS}(\sigma_r), \tau) + \sum_{i=1}^{m} h_{OS}^{(i)}(\tau). \quad (1)$$

Let t_F be a time after the end of the input. Since DS does not hold a packet, $\sum_{i=1}^{m} h_{DS}^{(i)}(t_F) = 0$. Also, clearly $\sum_{i=1}^{m} h_{OS}^{(i)}(t_F) \geq 0$. On the other hand, $R_{AS}(\sigma_r, t_F) = R_{AS}(\sigma_r)$, $R_{OS}(\sigma_r, t_F) = R_{OS}(\sigma_r)$ and $T_{OS}(g_{OS}(\sigma_r), t_F) = T_{OS}(g_{OS}(\sigma_r))$. By the above argument and Eq. (1), $R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \geq T_{OS}(g_{OS}(\sigma_r))$.

We prove Eq. (1) by induction on time. At a time just before the first event, the statement is true because no algorithm holds a packet. Let t be an event time on σ_r. We assume that the statement is true at time t_- and show that it is true at t_+.

Case 1: t is an arrival event time: Let p be a packet which arrives at $Q^{(j)}$ at t. Of course, since no algorithm transmits a packet, $R_{AS}(\sigma_r, t_+) = R_{AS}(\sigma_r, t_-)$, $R_{OS}(\sigma_r, t_+) = R_{OS}(\sigma_r, t_-)$, and $T_{OS}(g_{OS}(\sigma_r), t_+) = T_{OS}(g_{OS}(\sigma_r), t_-)$. If OS rejects p, $h_{OS}^{(j)}(t_+) = h_{OS}^{(j)}(t_-)(= B)$. Hence, Eq. (1) is true by the induction hypothesis because the state of queues except for $Q^{(j)}$ does not change. If OS accepts p, $h_{OS}^{(j)}(t_+) = h_{OS}^{(j)}(t_-)+1$, which means $h_{OS}^{(j)}(t_-) \leq B-1$. Since $B-1 \geq h_{OS}^{(j)}(t_-) \geq h_{DS}^{(j)}(t_-)$ by Lemma 3.3, DS also accepts p. Hence, $h_{DS}^{(j)}(t_+) = h_{DS}^{(j)}(t_-) + 1$ holds. Since the state of queues except for $Q^{(j)}$ does not change, Eq. (1) is true by the induction hypothesis.

Case 2: t is a scheduling event time and DS executes Case AS: DS stores at least one α-packet at t_- by the condition of Case AS, and DS transmits a packet at t, that is, it is returned to DS in Case AS at t. Hence, $\sum_{i=1}^{m} h_{DS}^{(i)}(t_+) = \sum_{i=1}^{m} h_{DS}^{(i)}(t_-) - 1$ and $R_{AS}(\sigma_r, t_+) = R_{AS}(\sigma_r, t_-) + 1$.

Furthermore, OS is not executed because this scheduling event at t does not exist in the inputs $f_{OS}(\sigma_r)$ and $g_{OS}(\sigma_r)$ by the way of constructing $f_{OS}(\sigma_r)$. Thus, $T_{OS}(g_{OS}(\sigma_r), t_+) =$

$T_{OS}(g_{OS}(\sigma_r), t_-)$, $R_{OS}(\sigma_r, t_+) = R_{OS}(\sigma_r, t_-)$, and $\sum_{i=1}^{m} h_{OS}^{(i)}(t_+) = \sum_{i=1}^{m} h_{OS}^{(i)}(t_-)$. By the above equalities and the induction hypothesis, Eq. (1) is true.

Case 3: t is a scheduling event time and DS executes Case OS: Since DS does not hold an α-packet at t_- by the condition of Case OS, $R_{AS}(\sigma_r, t_+) = R_{AS}(\sigma_r, t_-)$. Let x_i be the number of packets p' such that OS transmits p' from $Q^{(i)}$ at t and DS does not store $X(p')$ in its buffer at t_-. Moreover, let $y_i \in \{0, 1\}$ be the number of 1-packets p'' such that OS returns p'' from $Q^{(i)}$ at t and DS stores $X(p'')$ in its buffer at t_-. Note that $\sum_{i=1}^{m} y_i \leq 1$ (The definitions of x_i and y_i are the same as those used in the proof of Lemma 3.3). Then,

$$\sum_{i=1}^{m} h_{OS}^{(i)}(t_+) = \sum_{i=1}^{m} (h_{OS}^{(i)}(t_-) - x_i - y_i)$$

holds. Furthermore, since OS also transmits a packet returned to DS,

$$T_{OS}(g_{OS}(\sigma_r), t_+) = T_{OS}(g_{OS}(\sigma_r), t_-) + \sum_{i=1}^{m} (x_i + y_i).$$

On the other hand, since DS transmits a packet when OS returns a packet,

$$\sum_{i=1}^{m} h_{DS}^{(i)}(t_+) = \sum_{i=1}^{m} (h_{DS}^{(i)}(t_-) - y_i)$$

and

$$R_{OS}(\sigma_r, t_+) = R_{OS}(\sigma_r, t_-) + \sum_{i=1}^{m} y_i.$$

Therefore, Eq. (1) is true by the above equalities and the induction hypothesis. □

4 Competitive Analysis of *DS*

In this section, we prove the following theorem:

Theorem 4.1 *The competitive ratio of DS is at most* $c + \frac{2-c}{\alpha(2-c)+c-1}$.

The rest of this section is devoted to the proof of Theorem 4.1.

4.1 Bounding the Numbers of α-packets

Let σ_r for M_r be an input for DS. Let $\mathcal{T}_{B,1}(\sigma_r)$ ($\mathcal{T}_{B,\alpha}(\sigma_r)$, respectively) be the number of 1-packets (α-packets, respectively) p such that (i) just before p arrives, its destination buffer of DS stores B α-packets, (ii) DS does not transmit p but OPT_r transmits p. Also, let $\mathcal{T}_{\overline{B},1}(\sigma_r)$ ($\mathcal{T}_{\overline{B},\alpha}(\sigma_r)$, respectively) be the number of 1-packets (α-packets, respectively) p such that (i) just before p arrives, its destination buffer of DS stores at most $B-1$ α-packets, (ii) DS does not transmit p but OPT_r transmits p. We first show that $\mathcal{T}_{\overline{B},\alpha}(\sigma_r) = 0$.

Lemma 4.2 $\mathcal{T}_{\overline{B},\alpha}(\sigma_r) = 0$.

Proof. DS accepts arriving α-packets greedily, namely, it accepts an arriving α-packet whenever its destination buffer has at most $B-1$ α-packets. Since DS never preempts an α-packet as we have discussed in Sec. 3.1, those α-packets are eventually transmitted. Thus, Condition (ii) of the definition of $\mathcal{T}_{\overline{B},\alpha}(\sigma_r)$ fails and hence, $\mathcal{T}_{\overline{B},\alpha}(\sigma_r) = 0$ holds. □

The next lemma shows that we can assume without loss of generality that $\mathcal{T}_{B,1}(\sigma_r) = 0$.

Lemma 4.3 *For any input σ_r for M_r, there exists an input σ'_r for M_r such that $\mathcal{T}_{B,1}(\sigma'_r) = 0$ and $\frac{V_{OPT_r}(\sigma'_r)}{V_{DS}(\sigma'_r)} \geq \frac{V_{OPT_r}(\sigma_r)}{V_{DS}(\sigma_r)}$.*

Proof. If $\mathcal{T}_{B,1}(\sigma_r) = 0$, then we are done (simply let $\sigma'_r = \sigma_r$). If $\mathcal{T}_{B,1}(\sigma_r) > 0$, then let p_j ($j = 1, \ldots, \mathcal{T}_{B,1}(\sigma_r)$) be a 1-packet satisfying the condition of $\mathcal{T}_{B,1}(\sigma_r)$. Then, construct an input σ'_r from σ_r by replacing each 1-packet p_j with an α-packet q_j. Since at each arrival of q_j, DS's corresponding buffer is full of α-packets, DS rejects q_j. Therefore, the behavior of DS is exactly the same for σ_r and σ'_r, thus $V_{DS}(\sigma'_r) = V_{DS}(\sigma_r)$. We can define an offline algorithm OFF for σ'_r whose behavior is exactly the same as that of OPT_r for σ_r. Then since OFF transmits q_j for every j, $V_{OPT_r}(\sigma'_r) \geq V_{OFF}(\sigma'_r) \geq V_{OPT_r}(\sigma_r) + \mathcal{T}_{B,1}(\sigma_r)(\alpha - 1) \geq V_{OPT_r}(\sigma_r)$. This completes the proof. □

By Lemmas 4.2 and 4.3, the numbers of 1-packets and α-packets respectively that OPT_r transmits but DS does not are $\mathcal{T}_{\overline{B},1}(\sigma_r)$ and $\mathcal{T}_{B,\alpha}(\sigma_r)$. Let $\overline{\mathcal{T}}_1(\sigma_r)$ ($\overline{\mathcal{T}}_\alpha(\sigma_r)$) denote the number of 1-packets (α-packets) which DS transmits but OPT_r does not. Also, let $V_{common}(\sigma_r)$ denote the total value of packets transmitted by both DS and OPT_r. Then,

$$V_{OPT_r}(\sigma_r) = V_{common}(\sigma_r) + \mathcal{T}_{\overline{B},1}(\sigma_r) + \alpha\mathcal{T}_{B,\alpha}(\sigma_r)$$

and

$$V_{DS}(\sigma_r) = V_{common}(\sigma_r) + \overline{\mathcal{T}}_1(\sigma_r) + \alpha\overline{\mathcal{T}}_\alpha(\sigma_r).$$

By these equalities,

$$V_{OPT_r}(\sigma_r) = V_{DS}(\sigma_r) - \overline{\mathcal{T}}_1(\sigma_r) - \alpha\overline{\mathcal{T}}_\alpha(\sigma_r) + \mathcal{T}_{\overline{B},1}(\sigma_r) + \alpha\mathcal{T}_{B,\alpha}(\sigma_r). \tag{2}$$

By the definitions of $R_{AS}(\sigma_r)$ and $R_{OS}(\sigma_r)$,

$$V_{DS}(\sigma_r) = \alpha R_{AS}(\sigma_r) + R_{OS}(\sigma_r). \tag{3}$$

The next lemma bounds the difference between the number of α-packets transmitted by OPT_r and DS using those transmitted by AS.

Lemma 4.4

$$\mathcal{T}_{B,\alpha}(\sigma_r) - \overline{\mathcal{T}}_\alpha(\sigma_r) \leq (c-1)R_{AS}(\sigma_r). \tag{4}$$

Proof. As stated in Case 2 of the proof of Lemma 3.3, DS does not execute Case AS1.2. Hence, we have

$$T_{AS}(g_{AS}(\sigma_r)) = R_{AS}(\sigma_r) \tag{5}$$

and

$$g_{AS}(\sigma_r) = f_{AS}(\sigma_r). \tag{6}$$

By the definitions of c and AS (i.e. A),

$$cT_{AS}(f_{AS}(\sigma_r)) \geq T_{OPT_1}(f_{AS}(\sigma_r)). \tag{7}$$

Let $OPT_{r,\alpha}$ be the offline algorithm which accepts and transmits only α-packets transmitted by OPT_r. Using Lemma 4.2, we have

$$T_{OPT_1}(f_{AS}(\sigma_r)) \geq T_{OPT_{r,\alpha}}(\sigma_r) = \mathcal{T}_{\overline{B},\alpha}(\sigma_r) + \mathcal{T}_{B,\alpha}(\sigma_r) + T_{common} = \mathcal{T}_{B,\alpha}(\sigma_r) + T_{common}, \tag{8}$$

where T_{common} denotes the number of α-packets which both OPT_r and DS transmit, and the inequality follows from the optimality of OPT_1. Also, we have

$$R_{AS}(\sigma_r) = \overline{\mathcal{T}}_{\alpha}(\sigma_r) + T_{common}. \tag{9}$$

Then,

$$\begin{aligned} cR_{AS}(\sigma_r) &= cT_{AS}(g_{AS}(\sigma_r)) = cT_{AS}(f_{AS}(\sigma_r)) && \text{(by Eqs. (5) and (6))} \\ &\geq T_{OPT_1}(f_{AS}(\sigma_r)) \geq \mathcal{T}_{B,\alpha}(\sigma_r) + T_{common} && \text{(by Eqs. (7) and (8))} \\ &= \mathcal{T}_{B,\alpha}(\sigma_r) + R_{AS}(\sigma_r) - \overline{\mathcal{T}}_{\alpha}(\sigma_r). && \text{(by Eq. (9))} \end{aligned}$$

The proof can be completed by rearranging this inequality. □

4.2 Free Cells, p-events and f-events

Our next task is to evaluate the number of 1-packets which DS or OPT_r transmits, that is, to bound the value $\mathcal{T}_{\overline{B},1}(\sigma_r) - \overline{\mathcal{T}}_1(\sigma_r)$ from above. As we have seen in Lemma 4.4, bounding $\mathcal{T}_{B,\alpha}(\sigma_r) - \overline{\mathcal{T}}_{\alpha}(\sigma_r)$ was relatively easy because $g_{AS}(\sigma_r) = f_{AS}(\sigma_r)$. However, since $g_{OS}(\sigma_r) \neq f_{OS}(\sigma_r)$, it is not very easy for 1-packets. To overcome this difficulty, in this section we introduce useful tools for analysis, *free cells*, *p-events*, *f-events*, *$\overline{p}$-events*, and *$\overline{f}$-events*.

Recall that the size of the buffer is B. We think that a buffer consists of B *cells*, each of which can store one packet. Suppose that σ is an input in $\{\sigma_r, g_{OS}(\sigma_r)\}$, and OFF is an offline algorithm for σ which greedily accepts arriving packets and can preempt a packet only when its buffer is full. At a non-event time t, suppose that $h^{(i)}_{OS}(t) - h^{(i)}_{OFF}(t) \geq 0$. Then, *the number of OFF's free cells in $Q^{(i)}$ at t* is defined as $h^{(i)}_{OS}(t) - h^{(i)}_{OFF}(t)$. Similarly, if $h^{(i)}_{OFF}(t) - h^{(i)}_{OS}(t) \geq 0$, then *the number of OS's free cells in $Q^{(i)}$ at t* is $h^{(i)}_{OFF}(t) - h^{(i)}_{OS}(t)$.

We call an arrival event at which the number of OFF's free cells decreases a *profit-event* (or a *p-event* for short) for (OFF, σ). We will omit "for (OFF, σ)" when it is clear from the context. Observe that a p-event occurs when OFF accepts the arriving packet without preemption, and OS rejects it. Note that at the beginning of the input, the number of OFF's free cells is 0 for every queue. Thus, if a p-event occurs at a time t, then there must be an event before t which increases the number of OFF's free cells. We call such an event a *free cell-event* (or an *f-event* for short). Recall that both OS and OFF accept arriving packets greedily and OFF preempts a packet only when the buffer is full. Therefore, an arrival event cannot increase the number of OFF's free cells, and hence an f-event must be a scheduling event. We give a formal definition. For a p-event e_a at $Q^{(i)}$ at a time t, the *f-event corresponding to e_a* is a scheduling event e_s which occurs at $Q^{(i)}$ at a time $t' < t$ satisfying the following three conditions: (i) At e_s, OFF transmits a packet from $Q^{(i)}$ but OS does not transmit a packet from $Q^{(i)}$, (ii) the number of OFF's free cells in $Q^{(i)}$ is $h^{(i)}_{OS}(t_-) - h^{(i)}_{OFF}(t_-) - 1$ just before e_s and $h^{(i)}_{OS}(t_-) - h^{(i)}_{OFF}(t_-)$ just after e_s, and (iii) after e_s, the

number of OFF's free cells in $Q^{(i)}$ retains at least $h^{(i)}_{OS}(t_-) - h^{(i)}_{OFF}(t_-)$ until e_a. Note that for each p-event, the corresponding f-event is uniquely determined and this mapping is an injection. An f-event is called an *f-event for* (OFF, σ) if the corresponding p-event is for (OFF, σ).

Similarly, we call an arrival event at which the number of OS's free cells decreases an *anti profit-event* (or an *$\overline{p}$-event*) for (OFF, σ). Observe that an $\overline{p}$-event occurs when OFF either rejects the arriving packet or preempts a packet and accepts it, and OS accepts it. For the same reason as above, for each $\overline{p}$-event, there must be a scheduling event that causes increase in the number of OS's free cells. We call such an event an *anti free cell-event* (or an *$\overline{f}$-event*). Formally, for an $\overline{p}$-event $\overline{e}_a$ at $Q^{(i)}$ at a time t, the corresponding *$\overline{f}$-event* is a scheduling event $\overline{e}_s$ which occurs at $Q^{(i)}$ at a time $t' < t$ satisfying the following three conditions: (i) At $\overline{e}_s$, OS transmits a packet from $Q^{(i)}$ but OFF does not transmit a packet from $Q^{(i)}$, (ii) the number of OS's free cells in $Q^{(i)}$ is $h^{(i)}_{OFF}(t_-) - h^{(i)}_{OS}(t_-) - 1$ just before $\overline{e}_s$ and $h^{(i)}_{OFF}(t_-) - h^{(i)}_{OS}(t_-)$ just after $\overline{e}_s$, and (iii) after $\overline{e}_s$, the number of OS's free cells in $Q^{(i)}$ retains at least $h^{(i)}_{OFF}(t_-) - h^{(i)}_{OS}(t_-)$ until $\overline{e}_a$. Similarly to the case for p-events and f-events, this mapping is an injection. An $\overline{f}$-event is called an *$\overline{f}$-event for* (OFF, σ) if the corresponding $\overline{p}$-event is for (OFF, σ).

Here, we give some definitions on the numbers of p-events, $\overline{p}$-events, f-events and $\overline{f}$-events. For an input $\sigma \in \{\sigma_r, g_{OS}(\sigma_r)\}$ and an offline algorithm OFF for σ which greedily accepts arriving packets and preempts a packet only when its buffer is full, let $\mathcal{P}_{OFF}(\sigma)$ ($\overline{\mathcal{P}}_{OFF}(\sigma)$) denote the number of p-events ($\overline{p}$-events) for (OFF, σ). Also, let $\mathcal{F}_{OFF}(\sigma)$ ($\overline{\mathcal{F}}_{OFF}(\sigma)$) denote the number of f-events ($\overline{f}$-events) for (OFF, σ). First we prove a simple but important lemma on these events.

Lemma 4.5 *For any input $\sigma \in \{\sigma_r, g_{OS}(\sigma_r)\}$ and any offline algorithm OFF for σ which greedily accepts arriving packets and can preempt a packet only when its buffer is full, $\mathcal{P}_{OFF}(\sigma) = \mathcal{F}_{OFF}(\sigma)$ and $\overline{\mathcal{P}}_{OFF}(\sigma) = \overline{\mathcal{F}}_{OFF}(\sigma)$.*

Proof. Since there exists a one-to-one mapping between p-events and f-events for each queue $Q^{(i)}$, the number of p-events is equal to that of f-events. Therefore, the total number of p-events is the same as that of f-events. Similarly, the total number of $\overline{p}$-events is equal to that of $\overline{f}$-events. □

4.3 Bounding the Numbers of 1-packets

In this section, we bound the number of 1-packets transmitted by DS or OPT_r, and finally completes the proof of Theorem 4.1. The first lemma to be proven is the following:

Lemma 4.6 $\mathcal{T}_{B,\alpha}(\sigma_r) + \mathcal{T}_{\overline{B},1}(\sigma_r) - \overline{\mathcal{T}}_{\alpha}(\sigma_r) - \overline{\mathcal{T}}_{1}(\sigma_r) \leq \mathcal{P}_{OPT_r}(\sigma_r) - \overline{\mathcal{P}}_{OPT_r}(\sigma_r)$.

Proof. We give a few definitions to prove this lemma. We say that an algorithm *drops* a packet if the algorithm rejects or preempts the packet. That is, if the algorithm does not drop a packet which has already arrived, it has transmitted the packet or the packet exists in its buffer.

For a non-event time t' on σ_r, $\mathcal{T}_{\overline{B},1}(\sigma_r, t')$ denotes the number of 1-packets p such that (i) p arrives at a time $t'' < t'$, (ii) its destination buffer of DS stores at most $B-1$ α-packets at t''_-, and (iii) DS drops p before t' but OPT_r does not drop p before t'.

Let $\mathcal{T}_{B,\alpha}(\sigma_r, t')$ be the number of α-packets p such that (i) p arrives at a time $t'' < t'$, (ii) its destination buffer of DS stores B α-packets at t''_-, and (iii) DS rejects p at t'' but OPT_r does not drop p before t'.

$\overline{\mathcal{T}}_{\alpha}(\sigma_r, t')$ ($\overline{\mathcal{T}}_{1}(\sigma_r, t')$) denotes the number of α-packets (1-packets) p such that (i) p arrives at a time $t'' < t'$, and (ii) DS does not drop p before t' but OPT_r drops p before t'.

Time	Event	Action	OS	OFF	Free cells
0			0	0	0
1	arrival	A packet arrives	1	1	0
2	arrival	A packet arrives	2	2	0
3	arrival	A packet arrives	3	3	0
4	arrival	A packet arrives	4	4	0
5	scheduling	Only OS schedules	3	4	0
6	scheduling	Only OFF schedules	3	3	0
7	scheduling	Only OFF schedules	3	2	1
8	arrival	A packet arrives	4	3	1
9	scheduling	Only OFF schedules	4	2	2
10	scheduling	Only OFF schedules	4	1	3
11	arrival	A packet arrives	4	2	2
12	scheduling	Only OS schedules	3	2	1
13	arrival	A packet arrives	4	3	1
14	scheduling	Only OFF schedules	4	2	2
15	arrival	A packet arrives	4	3	1
16	arrival	A packet arrives	4	4	0
17	scheduling	Only OFF schedules	4	3	1
18	scheduling	OFF and OS schedule	3	2	1
19	scheduling	OFF and OS schedule	2	1	1
20	scheduling	Only OS schedules	1	1	0
21	scheduling	OFF and OS schedule	0	0	0

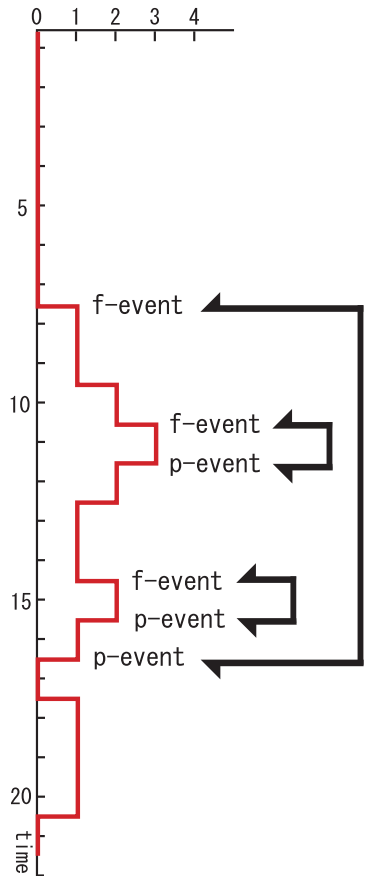

Figure 2: We give an example to explain p-events and f-events occurring at $Q^{(i)}$ for $B = 4$. Time, Event and Action columns show times at which events occur at $Q^{(i)}$, the names of the events, and OS's and OFF's actions for the events, respectively. OS and OFF columns show the numbers of OS's and OFF's packets in $Q^{(i)}$ at each time, respectively. Free cells column shows the number of OFF's free cells at each time. For example, at time 11, OS rejects an arriving packet but OFF accepts it without preemption. Then, the number of OFF's free cells decreases and hence this arrival event is a p-event. The f-event corresponding to this p-event occurs at time 10, at which OFF transmits a packet from $Q^{(i)}$ but OS does not. It is easy to see that p-events and f-events form balanced parenthesis.

For any input $\sigma \in \{\sigma_r, g_{OS}(\sigma_r)\}$, any offline algorithm OFF for σ which greedily accepts arriving packets and can preempt a packet only when its buffer is full, and a non-event time t' on σ, $\mathcal{P}_{OFF}(\sigma, t')$ ($\overline{\mathcal{P}}_{OFF}(\sigma, t')$) denotes the number of p-events ($\overline{p}$-events) for (OFF, σ) which occur before t'.

Now we prove by induction on time that for a non-event time t' on σ_r,

$$\mathcal{T}_{B,\alpha}(\sigma_r, t') + \mathcal{T}_{\overline{B},1}(\sigma_r, t') - \overline{\mathcal{T}}_{\alpha}(\sigma_r, t') - \overline{\mathcal{T}}_{1}(\sigma_r, t') \leq \mathcal{P}_{OPT_r}(\sigma_r, t') - \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t'). \quad (10)$$

The statement of the lemma is immediate by taking t' as a time after the end of the input.

At a time just before the first event in σ_r, the statement is true because no packet has arrived yet and hence all the terms are 0. Let t be an event time. We assume that the statement is true at time t_- and show that it is true at t_+.

Since a p-event ($\overline{p}$-event) occurs at only an arrival event, its number does not change at a scheduling event. By definition, the left side of Eq. (10) does not change at a scheduling event. Thus, by the induction hypothesis the statement holds.

Second we consider the case in which an arrival event occurs, which is categorized into the following nine cases. Suppose that a packet p arrives at $Q^{(i)}$ at t.

Case 1: OPT_r accepts p without preemption: An arrival event at which OPT_r accepts the arriving packet is not an $\overline{p}$-event by definition. Thus, $\overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_+) = \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_-)$.

Case 1.1: DS accepts p without preemption: Since both DS and OPT_r accept p, the left side of Eq. (10) does not change. On the other hand, $\mathcal{P}_{OPT_r}(\sigma_r, t_-)$ does not decrease, which means that $\mathcal{P}_{OPT_r}(\sigma_r, t_+) \geq \mathcal{P}_{OPT_r}(\sigma_r, t_-)$. By the above argument and the induction hypothesis, Eq. (10) is true at t_+.

Case 1.2: DS rejects p: Since DS rejects p, $\overline{\mathcal{T}}_{\alpha}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_{1}(\sigma_r, t_-)$ do not change. Of course, p is a 1-packet or an α-packet, and $\mathcal{T}_{B,\alpha}(\sigma_r, t_+) + \mathcal{T}_{\overline{B},1}(\sigma_r, t_+) = \mathcal{T}_{B,\alpha}(\sigma_r, t_-) + \mathcal{T}_{\overline{B},1}(\sigma_r, t_-) + 1$ by definition. Also, by the condition of Case 1.2, DS rejects p, which means $h^{(i)}_{DS}(t_-) = B$. Hence, $h^{(i)}_{OS}(t_-) \geq h^{(i)}_{DS}(t_-) = B$ by Lemma 3.3. Then, OS rejects p. That is, $\mathcal{P}_{OPT_r}(\sigma_r, t_+) = \mathcal{P}_{OPT_r}(\sigma_r, t_-) + 1$. The above argument and the induction hypothesis show that Eq. (10) is true.

Case 1.3: DS preempts a packet p' and accepts p: By the greediness of DS, p and p' are an α-packet and a 1-packet, respectively. Thus, $\mathcal{T}_{B,\alpha}(\sigma_r, t_+) = \mathcal{T}_{B,\alpha}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_{\alpha}(\sigma_r, t_+) = \overline{\mathcal{T}}_{\alpha}(\sigma_r, t_-)$. If OPT_r does not drop p' before t_+, i.e., p' exists in its buffer at t_+ or OPT_r transmits p' before t_+, then $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-) + 1$ and $\overline{\mathcal{T}}_{1}(\sigma_r, t_+) = \overline{\mathcal{T}}_{1}(\sigma_r, t_-)$. If OPT_r drops p' before t_+, i.e., OPT_r rejects or preempts p' before t_+, then $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_{1}(\sigma_r, t_+) = \overline{\mathcal{T}}_{1}(\sigma_r, t_-) - 1$. That is, in either case $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) - \overline{\mathcal{T}}_{1}(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-) - \overline{\mathcal{T}}_{1}(\sigma_r, t_-) + 1$ holds. Moreover, since DS preempts p', $h^{(i)}_{DS}(t_-) = B$. Then, $h^{(i)}_{OS}(t_-) \geq h^{(i)}_{DS}(t_-) = B$ by Lemma 3.3. Hence, OS rejects p. Thus, $\mathcal{P}_{OPT_r}(\sigma_r, t_+) = \mathcal{P}_{OPT_r}(\sigma_r, t_-) + 1$. Therefore, Eq. (10) is true by the above argument and the induction hypothesis.

Case 2: OPT_r rejects p: Since a p-event does not occur in this case, $\mathcal{P}_{OPT_r}(\sigma_r, t_+) = \mathcal{P}_{OPT_r}(\sigma_r, t_-)$.

Case 2.1: DS accepts p without preemption: $\mathcal{T}_{B,\alpha}(\sigma_r, t_-) + \mathcal{T}_{\overline{B},1}(\sigma_r, t_-)$ does not change because DS accepts p which OPT_r rejects. Also, $\overline{\mathcal{T}}_{\alpha}(\sigma_r, t_+) + \overline{\mathcal{T}}_{1}(\sigma_r, t_+) = \overline{\mathcal{T}}_{\alpha}(\sigma_r, t_-) + \overline{\mathcal{T}}_{1}(\sigma_r, t_-) + 1$. Moreover, $\overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_+) \geq \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_-) + 1$ because OS may accept p. Thus, the above argument together with the induction hypothesis shows that Eq. (10) is true.

Case 2.2: DS rejects p: Since both OPT_r and DS reject p, the left side of Eq. (10) does not change. On the other hand, we can show that OS rejects p in the same way as Case 1.2 which is stated above. Hence, $\overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_+) = \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_-)$. Since no value in Eq. (10) changes, Eq. (10)

holds by the induction hypothesis.

Case 2.3: *DS* preempts a packet p' and accepts p: By the definition of DS, p is an α-packet and p' is a 1-packet. Hence, $\overline{\mathcal{T}}_\alpha(\sigma_r, t_+) = \overline{\mathcal{T}}_\alpha(\sigma_r, t_-) + 1$ and $\mathcal{T}_{B,\alpha}(\sigma_r, t_+) = \mathcal{T}_{B,\alpha}(\sigma_r, t_-)$. Also, we can show $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) - \overline{\mathcal{T}}_1(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-) - \overline{\mathcal{T}}_1(\sigma_r, t_-) + 1$ in the same way as Case 1.3. Therefore, the left side of Eq. (10) does not change. On the other hand, OS rejects p, which can be shown in the same way as Case 1.3 as well. Thus, $\overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_+) = \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_-)$. By the above argument and the induction hypothesis, Eq. (10) is true.

Case 3: OPT_r preempts a packet p'' and accepts p: Since a p-event does not occur in this case by definition, $\mathcal{P}_{OPT_r}(\sigma_r, t_+) = \mathcal{P}_{OPT_r}(\sigma_r, t_-)$.

Case 3.1: *DS* accepts p without preemption: Since p is an α-packet, $\mathcal{T}_{B,\alpha}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_\alpha(\sigma_r, t_-)$ do not change. Also, since p'' is a 1-packet, this case is similar to Case 1.3, in which DS preempts a packet. If DS does not drop p'' before t_+, then $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_1(\sigma_r, t_+) = \overline{\mathcal{T}}_1(\sigma_r, t_-) + 1$. If DS drops p'' before t_+, then $\overline{\mathcal{T}}_1(\sigma_r, t_+) = \overline{\mathcal{T}}_1(\sigma_r, t_-)$. In addition, we assume that $\mathcal{T}_{B,1}(\sigma_r) = 0$ by Lemma 4.3, and in the case in which DS rejects a 1-packet, the number of α-packets in its destination buffer is at most $B-1$. Thus, $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-) - 1$. That is, in either case, $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) - \overline{\mathcal{T}}_1(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-) - \overline{\mathcal{T}}_1(\sigma_r, t_-) - 1$. Furthermore, OS may accept p and hence $\overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_+) \geq \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_-) + 1$. By the above argument, the right side of Eq. (10) may decrease by one and the left side certainly decreases by one. Therefore, Eq. (10) is true by the induction hypothesis.

Case 3.2: *DS* rejects p: $\overline{\mathcal{T}}_\alpha(\sigma_r, t_-)$ does not change because p is an α-packet. Also, $\mathcal{T}_{B,\alpha}(\sigma_r, t_+) = \mathcal{T}_{B,\alpha}(\sigma_r, t_-) + 1$. Since $h^{(i)}_{OS}(t_-) \geq h^{(i)}_{DS}(t_-) = B$ by Lemma 3.3, OS rejects p and thus $\overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_+) = \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_-)$. In regard to p'' in the same way as Case 3.1, $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) - \overline{\mathcal{T}}_1(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-) - \overline{\mathcal{T}}_1(\sigma_r, t_-) - 1$. By the above, the values in the right side of Eq. (10) do not change, and the total value of the left side does not change. Hence, the induction hypothesis together with the above argument shows Eq. (10) holds.

Case 3.3: *DS* preempts a packet p' and accepts p: OS rejects p because $h^{(i)}_{OS}(t_-) \geq h^{(i)}_{DS}(t_-) = B$ by Lemma 3.3. Thus, $\overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_+) = \overline{\mathcal{P}}_{OPT_r}(\sigma_r, t_-)$. Since both DS and OPT_r accept p, which is an α-packet, $\mathcal{T}_{B,\alpha}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_\alpha(\sigma_r, t_-)$ do not change. If $p' = p''$, $\mathcal{T}_{\overline{B},1}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_1(\sigma_r, t_-)$ do not change. If $p' \neq p''$, we consider p' and p'' in the same way as Cases 1.3 and 3.1 respectively, and can show the total value of the left side of Eq. (10) does not change. For example, if OPT_r drops p' before t_+ and DS does not drop p'' before t_+, then $\mathcal{T}_{\overline{B},1}(\sigma_r, t_+) = \mathcal{T}_{\overline{B},1}(\sigma_r, t_-)$ and $\overline{\mathcal{T}}_1(\sigma_r, t_+) = \overline{\mathcal{T}}_1(\sigma_r, t_-)$. By the above argument, the values in the right side do not change and the total value of the left side does not change. Therefore, Eq. (10) is true by the induction hypothesis. □

Lemma 4.7

$$\mathcal{F}_{OPT_r}(\sigma_r) - \overline{\mathcal{F}}_{OPT_r}(\sigma_r) \leq R_{AS}(\sigma_r) + (c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) \tag{11}$$

and

$$\mathcal{F}_{OPT_r}(\sigma_r) - \overline{\mathcal{F}}_{OPT_r}(\sigma_r) \leq R_{AS}(\sigma_r) + R_{OS}(\sigma_r). \tag{12}$$

Proof. For input σ_r for M_r, let k denote the number of executions of Case OS1.2 for σ_r, that is, the number of scheduling events added to $f_{OS}(\sigma_r)$ to construct $g_{OS}(\sigma_r)$. We call these additional scheduling events *OS-events*. Let k' denote the number of scheduling events removed from σ_r to construct $f_{OS}(\sigma_r)$. We call these removed scheduling events *OPT_r-events*. By definition

$$k = T_{OS}(g_{OS}(\sigma_r)) - R_{OS}(\sigma_r) \tag{13}$$

and

$$k' = R_{AS}(\sigma_r). \tag{14}$$

First we will show Eq. (11). We define the following offline algorithm A' for $g_{OS}(\sigma_r)$: A' greedily accepts packets. At a scheduling event, A' chooses the same queue as the one OPT_r for σ_r chooses, but A' does nothing if it is an OS-event. Recall that OPT_r is an offline algorithm which greedily accepts arriving packets and can preempt a packet only when its buffer is full. Hence, A' has the same property.

For $g_{OS}(\sigma_r)$, let $\overline{T}_{A'}(g_{OS}(\sigma_r))$ denote the number of packets which A' transmits but OS does not. Also, let $\overline{T}_{OS}(g_{OS}(\sigma_r))$ denote the number of packets which OS transmits but A' does not. In the same way as in the proof of Lemma 4.4, we can show

$$\overline{T}_{A'}(g_{OS}(\sigma_r)) - \overline{T}_{OS}(g_{OS}(\sigma_r)) \leq (c-1)T_{OS}(g_{OS}(\sigma_r)), \tag{15}$$

which follows from the fact that OS is c-competitive for M_1.

Now we consider the relation between the value of $\overline{T}_{A'}(g_{OS}(\sigma_r)) - \overline{T}_{OS}(g_{OS}(\sigma_r))$ and the numbers of f-events and $\overline{f}$-events. Suppose that, at a p-event for $(A', g_{OS}(\sigma_r))$, A' accepts and OS rejects an arriving packet. Then, let z_1 and z_2 denote the numbers of such packets transmitted and preempted, respectively, by A'. Similarly, suppose that, at an $\overline{p}$-event for $(A', g_{OS}(\sigma_r))$, OS accepts and A' rejects an arriving packet. Then, let w_1 denote the number of such packets transmitted by OS. Suppose that, at an arrival event, OS accepts a packet p and A' preempts a packet p' and accepts p. Then, let w_2 and w_3 denote the numbers of such packets p' transmitted and rejected, respectively, by OS.

By definition, $\overline{T}_{A'}(g_{OS}(\sigma_r)) = z_1$, $\overline{T}_{OS}(g_{OS}(\sigma_r)) = w_1 + w_2$, $\mathcal{P}_{A'}(g_{OS}(\sigma_r)) = \sum_{i=1}^{2} z_i$, $\overline{\mathcal{P}}_{A'}(g_{OS}(\sigma_r)) = \sum_{i=1}^{3} w_i$, and $z_2 = w_3$. By these equalities, we have

$$\mathcal{P}_{A'}(g_{OS}(\sigma_r)) - \overline{\mathcal{P}}_{A'}(g_{OS}(\sigma_r)) = \overline{T}_{A'}(g_{OS}(\sigma_r)) - \overline{T}_{OS}(g_{OS}(\sigma_r)). \tag{16}$$

By the definition of A', OPT_r for σ_r can accept an arriving packet at the arrival event in which A' for $g_{OS}(\sigma_r)$ accepts the packet. Hence, if an arrival event is a p-event for $(A', g_{OS}(\sigma_r))$, it is a p-event for (OPT_r, σ_r) as well by the definition of p-events. Also, if an arrival event is not an $\overline{p}$-event for $(A', g_{OS}(\sigma_r))$, then it is not an $\overline{p}$-event for (OPT_r, σ_r) either by the definition of $\overline{p}$-events. Let x' denote the number of arrival events which are p-events for (OPT_r, σ_r) but not for $(A', g_{OS}(\sigma_r))$. Similarly, let y' denote the number of arrival events which are $\overline{p}$-events for $(A', g_{OS}(\sigma_r))$ but not for (OPT_r, σ_r). Then, we have

$$\mathcal{P}_{OPT_r}(\sigma_r) = \mathcal{P}_{A'}(g_{OS}(\sigma_r)) + x' \tag{17}$$

and

$$\overline{\mathcal{P}}_{OPT_r}(\sigma_r) = \overline{\mathcal{P}}_{A'}(g_{OS}(\sigma_r)) - y'. \tag{18}$$

The number of arrival events at which OPT_r accepts the arriving packets without preemption and A' either rejects them or accepts them with preemption is $x' + y'$ by the definitions of p-events and $\overline{p}$-events. Therefore, OPT_r transmits $x' + y'$ more packets than A' does, but this is at most k' since OPT_r has k' more opportunities for transmission (i.e. OPT_r-events) than A'. (This can be seen as follows: Just imagine that at an OPT_r-event, if OPT_r transmits a packet then A' discards the same packet. Then the contents of buffers of OPT_r and A' are always the same, and OPT_r can transmit at most k' more packets.) Thus,

$$x' + y' \leq k'. \tag{19}$$

By the above argument,

$$\begin{aligned}
\mathcal{F}_{OPT_r}(\sigma_r) &- \overline{\mathcal{F}}_{OPT_r}(\sigma_r) = \mathcal{P}_{OPT_r}(\sigma_r) - \overline{\mathcal{P}}_{OPT_r}(\sigma_r) && \text{(by Lemma 4.5)}\\
&= \mathcal{P}_{A'}(g_{OS}(\sigma_r)) - \overline{\mathcal{P}}_{A'}(g_{OS}(\sigma_r)) + x' + y' && \text{(by Eqs. (17) and (18))}\\
&\le \mathcal{P}_{A'}(g_{OS}(\sigma_r)) - \overline{\mathcal{P}}_{A'}(g_{OS}(\sigma_r)) + k' && \text{(by Eq. (19))}\\
&\le \overline{T}_{A'}(g_{OS}(\sigma_r)) - \overline{T}_{OS}(g_{OS}(\sigma_r)) + k' && \text{(by Eq. (16))}\\
&\le (c-1)T_{OS}(g_{OS}(\sigma_r)) + R_{AS}(\sigma_r) && \text{(by Eqs. (15) and (14))}\\
&\le (c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) + R_{AS}(\sigma_r). && \text{(by Lemma 3.4)}
\end{aligned}$$

Second we will bound $\mathcal{F}_{OPT_r}(\sigma_r)$ from above to show Eq. (12). When a scheduling event is an f-event for (OPT_r, σ_r), OPT_r transmits a packet from some $Q^{(j)}$ at the scheduling event. In addition, OS must store at least one packet in $Q^{(j)}$. There exist $T_{OS}(g_{OS}(\sigma_r))$ scheduling events at which OS stores at least one packet, which include k OS-events but not k' OPT_r-events. Since only OS transmits packets at the k OS-events, they are not f-events. On the other hand, the k' OPT_r-events can be f-events. Hence, the number of f-events for (OPT_r, σ_r) is at most $T_{OS}(g_{OS}(\sigma_r)) + k' - k$. Therefore,

$$\begin{aligned}
\mathcal{F}_{OPT_r}(\sigma_r) &- \overline{\mathcal{F}}_{OPT_r}(\sigma_r) \le \mathcal{F}_{OPT_r}(\sigma_r) && \text{(by the fact } \overline{\mathcal{F}}_{OPT_r}(\sigma_r) \ge 0)\\
&\le T_{OS}(g_{OS}(\sigma_r)) + k' - k\\
&\le T_{OS}(g_{OS}(\sigma_r)) + R_{AS}(\sigma_r) - T_{OS}(g_{OS}(\sigma_r)) + R_{OS}(\sigma_r) && \text{(by Eqs. (13) and (14))}\\
&= R_{AS}(\sigma_r) + R_{OS}(\sigma_r).
\end{aligned}$$

□

Lemma 4.8

$$\mathcal{T}_{B,\alpha}(\sigma_r) + \mathcal{T}_{\overline{B},1}(\sigma_r) - \overline{\mathcal{T}}_{\alpha}(\sigma_r) - \overline{\mathcal{T}}_{1}(\sigma_r) \le R_{AS}(\sigma_r) + (c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) \tag{20}$$

and

$$\mathcal{T}_{B,\alpha}(\sigma_r) + \mathcal{T}_{\overline{B},1}(\sigma_r) - \overline{\mathcal{T}}_{\alpha}(\sigma_r) - \overline{\mathcal{T}}_{1}(\sigma_r) \le R_{AS}(\sigma_r) + R_{OS}(\sigma_r). \tag{21}$$

Proof. Using Lemmas 4.6 and 4.5, we have

$$\begin{aligned}
\mathcal{T}_{B,\alpha}(\sigma_r) + \mathcal{T}_{\overline{B},1}(\sigma_r) - \overline{\mathcal{T}}_{\alpha}(\sigma_r) - \overline{\mathcal{T}}_{1}(\sigma_r) &\le \mathcal{P}_{OPT_r}(\sigma_r) - \overline{\mathcal{P}}_{OPT_r}(\sigma_r)\\
&\le \mathcal{F}_{OPT_r}(\sigma_r) - \overline{\mathcal{F}}_{OPT_r}(\sigma_r).
\end{aligned}$$

This inequality and Lemma 4.7 directly imply Eqs. (20) and (21). □

We are now ready to complete the proof of Theorem 4.1. If $(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) \ge R_{OS}(\sigma_r)$ in Eqs. (20) and (21), then we have

$$R_{OS}(\sigma_r) \le \frac{c-1}{2-c} R_{AS}(\sigma_r). \tag{22}$$

Otherwise, we have

$$R_{AS}(\sigma_r) < \frac{2-c}{c-1} R_{OS}(\sigma_r). \tag{23}$$

Hence,

$$\begin{aligned}
V_{OPT_r}(\sigma_r) &= V_{DS}(\sigma_r) - \overline{\mathcal{T}}_1(\sigma_r) - \alpha\overline{\mathcal{T}}_\alpha(\sigma_r) + \mathcal{T}_{\overline{B},1}(\sigma_r) + \alpha\mathcal{T}_{B,\alpha}(\sigma_r) && \text{(by Eq. (2))} \\
&\le V_{DS}(\sigma_r) + R_{AS}(\sigma_r) + R_{OS}(\sigma_r) + (\alpha-1)(\mathcal{T}_{B,\alpha}(\sigma_r) - \overline{\mathcal{T}}_\alpha(\sigma_r)) && \text{(by Eq. (21))} \\
&\le V_{DS}(\sigma_r) + R_{AS}(\sigma_r) + R_{OS}(\sigma_r) + (\alpha-1)(c-1)R_{AS}(\sigma_r) && \text{(by Eq. (4))} \\
&= V_{DS}(\sigma_r) + (-\alpha+2+c\alpha-c)R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \\
&= V_{DS}(\sigma_r) + \frac{(-\alpha+2+c\alpha-c)R_{AS}(\sigma_r) + R_{OS}(\sigma_r)}{\alpha R_{AS}(\sigma_r) + R_{OS}(\sigma_r)} V_{DS}(\sigma_r). && \text{(by Eq. (3))} \qquad (24)
\end{aligned}$$

If $(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) \ge R_{OS}(\sigma_r)$, we have by Eqs. (22) and (24)

$$\begin{aligned}
V_{OPT_r}(\sigma_r) &\le V_{DS}(\sigma_r) + \frac{(-\alpha+2+c\alpha-c)R_{AS}(\sigma_r) + \frac{c-1}{2-c}R_{AS}(\sigma_r)}{\alpha R_{AS}(\sigma_r) + \frac{c-1}{2-c}R_{AS}(\sigma_r)} V_{DS}(\sigma_r) \\
&= \left(c + \frac{2-c}{\alpha(2-c)+c-1}\right) V_{DS}(\sigma_r).
\end{aligned}$$

Also, if $(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) < R_{OS}(\sigma_r)$, we have by Eqs. (23) and (24)

$$\begin{aligned}
V_{OPT_r}(\sigma_r) &\le V_{DS}(\sigma_r) + \frac{(-\alpha+2+c\alpha-c)\frac{2-c}{c-1}R_{OS}(\sigma_r) + R_{OS}(\sigma_r)}{\alpha\frac{2-c}{c-1}R_{OS}(\sigma_r) + R_{OS}(\sigma_r)} V_{DS}(\sigma_r) \\
&= \left(c + \frac{2-c}{\alpha(2-c)+c-1}\right) V_{DS}(\sigma_r).
\end{aligned}$$

This completes the proof of Theorem 4.1.

5 Competitive Ratios for 2-Value Multi-Queue Switch Models

In this section, we give upper bounds on several variants of M_2 using Theorem 4.1 and the competitive ratio of SS shown in Sec. 1. In the following results, for each value of B, we chose α maximizing the competitive ratio; to calculate the competitive ratios and the corresponding values of α, we used Mathematica.

5.1 Competitive Ratios for Arbitrary m

In this section, we use, as subroutines of DS and SS, the deterministic algorithm Semi-Greedy (SGR) for M_1 [37]. Its competitive ratios for several values of B are given in the second column of Table 3 as c. In the case of large enough B, the best known competitive ratio is $\frac{e}{e-1}$ achieved by $E^{M^{E\hat{P}'}}$ [8], and hence we use it as subroutines.

Corollary 5.1 *There is a deterministic online algorithm for the non-preemptive M_2 for large enough B and any m, whose competitive ratio is at most 3.177.*

Proof. There is a $(2-\frac{1}{\alpha})$-competitive non-preemptive deterministic algorithm for large enough B [7, 40] for the non-preemptive 2-value single queue model. By Azar and Richter [9] and our results, there is a deterministic online algorithm for the non-preemptive M_2 whose competitive ratio is at

Table 3: Competitive ratios of deterministic online algorithms for the preemptive M_2

B	c	α	Competitive Ratio
2	13/7	3.335	2.5596
3	35/18	3.647	2.5996
≥ 4 and even	17/9	3.453	2.5766
5	52/27	3.587	2.5931
7	23/12	3.554	2.5894
large enough	17/9	3.453	2.5766

The third column shows the values of α that maximize the competitive ratios.

most $\min\{c + \frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}(2 - \frac{1}{\alpha})(\triangleq d)$. Since an upper bound c on the competitive ratio of $E^{M^{\hat{EP'}}}$ is $\frac{e}{e-1}$ for large enough B, d takes the maximum value of

$$\frac{(-4+8e-6e^2+2e^3+e^{3/2}\sqrt{-2+4e-3e^2+e^3})(e^2+2\sqrt{-2e+4e^2-3e^3+e^4})}{(-1+e)(2-2e+e^2+\sqrt{-2e+4e^2-3e^3+e^4})(-e+e^2+\sqrt{-2e+4e^2-3e^3+e^4})} < 3.177$$

when $\alpha = \frac{2-2e+e^2+\sqrt{-2e+4e^2-3e^3+e^4}}{(-2+e)^2}$. □

Corollary 5.2 *There are deterministic online algorithms for the preemptive M_2 for any m, whose competitive ratios are shown in Table 3.*

Proof. There is a $\frac{-1-\alpha+2\alpha^2+\sqrt{1+2\alpha-3\alpha^2+4\alpha^3}}{2\alpha^2}(\triangleq d')$-competitive preemptive deterministic algorithm for any B for the preemptive 2-value single queue model [34]. By Azar and Richter [9] and our results, there is a deterministic online algorithm for the preemptive M_2 whose competitive ratio is at most $\min\{c + \frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}d'(\triangleq d)$. For each B, d takes the maximum value provided in the fourth column of Table 3 by α in the third column. □

For large enough B, we can obtain a better bound than 2.5766 shown in Table 3 using another preemptive 2-value single queue algorithm and $E^{M^{\hat{EP'}}}$ for M_1:

Corollary 5.3 *There is a deterministic online algorithm for the preemptive M_2 whose competitive ratio is at most* 2.516 *for large enough B and any m.*

Proof. There is a 1.282-competitive preemptive deterministic algorithm for the preemptive 2-value single queue model for large enough B [16]. Furthermore, an upper bound c on the competitive ratio of $E^{M^{\hat{EP'}}}$ is $\frac{e}{e-1}$ for large enough B. Therefore, the competitive ratio of a deterministic algorithm for the preemptive M_2 is at most $\min\{c+\frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}\cdot 1.282$ where $c = \frac{e}{e-1}$, which takes the maximum value of $\frac{e}{e-1} \cdot \frac{3e-e^2-\sqrt{-16e+33e^2-22e^3+5e^4}}{4e-2e^2} \cdot 1.282 < 2.516$ when $\alpha = \frac{3e-e^2-\sqrt{-16e+33e^2-22e^3+5e^4}}{4e-2e^2}$. □

Corollary 5.4 *There is a randomized online algorithm for the preemptive M_2 whose competitive ratio is at most* 2.270*for large enough B and any m.*

Proof. There is a $(1+\alpha^{-\frac{1}{2}}-\alpha^{-1})$-competitive preemptive randomized algorithm for the preemptive 2-value single queue model for large enough B [5]. An upper bound c on the competitive ratio of $E^{M^{\hat{EP'}}}$ is $\frac{e}{e-1}$ for large enough B. Therefore, the competitive ratio of a randomized algorithm for the preemptive M_2 is at most $\min\{c+\frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}(1+\alpha^{-\frac{1}{2}}-\alpha^{-1})$ where $c=\frac{e}{e-1}$, which takes the maximum value of 2.2690 when $\alpha = 2.2454$. □

5.2 Competitive Ratios for $m = 2$

In this section, we use, as subroutines of DS and SS, the deterministic algorithm Segmental Greedy (SG) for M_1 for $m = 2$ [32]. Its competitive ratios for several values of B are given in the second column of Table 4 as c. The values of c for $1 \le B \le 8$ are not explicitly written in the paper [32] but implied by its appendix.

Corollary 5.5 *There is a deterministic online algorithm for the non-preemptive M_2 for large enough B and $m = 2$, whose competitive ratio is at most 2.562.*

Proof. There is a $(2-\frac{1}{\alpha})$-competitive non-preemptive deterministic algorithm for large enough B [7, 40] for the non-preemptive 2-value single queue model. By Azar and Richter [9] and our results, there is a deterministic online algorithm for the non-preemptive M_2 whose competitive ratio is at most $\min\{c+\frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}(2-\frac{1}{\alpha})(\triangleq d)$. Since an upper bound c on the competitive ratio of SG is 16/13 for large enough B, d takes the maximum value of $4(1037-20\sqrt{2314})/117 < 2.562$ when $\alpha = (89+2\sqrt{2314})/50$. □

Corollary 5.6 *There are deterministic online algorithms for the preemptive M_2 for $m = 2$, whose competitive ratios are shown in Table 4.*

Proof. There is a $\frac{-1-\alpha+2\alpha^2+\sqrt{1+2\alpha-3\alpha^2+4\alpha^3}}{2\alpha^2}(\triangleq d')$-competitive preemptive deterministic algorithm for any B for the preemptive 2-value single queue model [34]. By Azar and Richter [9] and our results, there is a deterministic online algorithm for the preemptive M_2 whose competitive ratio is at most $\min\{c+\frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}d'(\triangleq d)$. For each value of B, d takes the maximum value provided in the fourth column of Table 4 by α in the third column. □

Corollary 5.7 *There is a randomized online algorithm for the preemptive M_2 whose competitive ratio is at most* 2.056 *for large enough B and $m = 2$.*

Proof. There is a $(1+\alpha^{-\frac{1}{2}}-\alpha^{-1})$-competitive preemptive randomized algorithm for the preemptive 2-value single queue model for large enough B [5]. An upper bound c on the competitive ratio of SG is 16/13 for large enough B. Therefore, the competitive ratio of a randomized algorithm for the preemptive M_2 is at most $\min\{c+\frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}(1+\alpha^{-\frac{1}{2}}-\alpha^{-1})$ where $c = 16/13$, which takes the maximum value of $4(-13+\sqrt{494}+2\sqrt{5(7+\sqrt{494})})/65 < 2.056$ when $\alpha = (7+\sqrt{494})/20$. □

Table 4: Competitive ratios of deterministic online algorithms for the preemptive M_2 when $m = 2$

B	c	α	Competitive Ratio
1	3/2	2.253	2.297
2, 3	4/3	1.853	2.169
4	13/10	1.776	2.145
5, 7	14/11	1.713	2.126
6	24/19	1.691	2.120
8	29/23	1.685	2.118
large enough	16/13	1.616	2.098

The third column shows the values of α that maximize the competitive ratios.

6 Concluding Remarks

In this paper, we have designed online algorithms DS and SS for the 2-value relaxed model. We have improved the previous competitive ratio of 2 to $\min\{c + \frac{2-c}{\alpha(2-c)+c-1}, c\alpha\}$, where c is the competitive ratio of an online algorithm for the unit-value multi-queue switch model, which is used as subroutines of DS and SS. As a result, we have improved competitive ratios for the 2-value multiqueue switch model in several settings.

We conclude the paper by providing open questions: (i) Our policy in this paper is to use an algorithm for M_1 for designing an algorithm for M_2. When determining acceptance/rejection of packets, we give absolute priority to α-packets (than 1-packets), which we think is a correct direction in our policy. However, we use greedy algorithms within α-packets and within 1-packets. Is there any elaborate way for buffer management of those packets (rather than simple greedy algorithms)? (ii) Our technique in this paper is available only for the 2-value case. Can it be extended to the multi-value case? (iii) The competitive analysis using the relaxed model may generate overhead, and hence may be inefficient. Can we construct better algorithms for M_2 without using the relaxed model?

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers 26730008 and 24500013.

References

[1] W. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosén, "Competitive queue policies for differentiated services," *Journal of Algorithms*, Vol. 55, No. 2, pp. 113–141, 2005.

[2] K. Al-Bawani, and A. Souza, "Buffer overflow management with class segregation," *Information Processing Letters*, Vol. 113, No. 4, pp. 145–150, 2013.

[3] K. Al-Bawani, M. Englert and M. Westermann, "Online Packet Scheduling for CIOQ and Buffered Crossbar Switches," *In Proc. of the 28th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 241–250, 2016.

[4] S. Albers and M. Schmidt, "On the performance of greedy algorithms in packet buffering," *SIAM Journal on Computing*, Vol. 35, No. 2, pp. 278–304, 2005.

[5] N. Andelman, "Randomized queue management for DiffServ," *In Proc. of the 17th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 1–10, 2005.

[6] N. Andelman and Y. Mansour, "Competitive management of non-preemptive queues with multiple values," *Distributed Computing*, Vol. 2848, pp. 166–180, 2003.

[7] N. Andelman, Y. Mansour and A. Zhu, "Competitive queueing policies for QoS switches," *In Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 761–770, 2003.

[8] Y. Azar and A. Litichevskey, "Maximizing throughput in multi-queue switches," *Algorithmica*, Vol.45, No. 1, pp, 69–90, 2006.

[9] Y. Azar and Y. Richter, "Management of multi-queue switches in QoS networks," *Algorithmica*, Vol.43, No. 1-2, pp, 81–96, 2005.

[10] Y. Azar and Y. Richter, "The zero-one principle for switching networks," *In Proc. of the 36th Annual ACM Symposium on Theory of Computing*, pp. 64–71, 2004.

[11] Y. Azar and Y. Richter, "An improved algorithm for CIOQ switches," *ACM Transactions on Algorithms*, Vol. 2, No. 2, pp. 282–295, 2006.

[12] M. Bienkowski and A. Madry, "Geometric aspects of online packet buffering: an optimal randomized algorithm for two buffers," *In Proc. of the 8th Latin American Theoretical INformatics*, pp. 252–263, 2008.

[13] M. Bienkowski, "An optimal lower bound for buffer management in multi-queue switches," *Algorithmica*, Vol.68, No.2, pp. 426–447, 2014.

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

[15] P. Chuprikov, S. I. Nikolenko and K. Kogan, "Priority queueing with multiple packet characteristics," *In Proc. of the 34th IEEE International Conference on Computer Communications*, pp. 1–9, 2015.

[16] M. Englert and M. Westermann, "Lower and upper bounds on FIFO buffer management in QoS switches," *In Proc. of the 14th Annual European Symposium on Algorithms*, pp. 352–363, 2006.

[17] P. Eugster , A. Kesselman, K. Kogan, S. I. Nikolenko, and A. Sirotkin, "Essential Traffic Parameters for Shared Memory Switch Performance," *In Proc. of the 22nd International Colloquium on Structural Information and Communication Complexity*, pp. 61–75, 2013.

[18] M. Goldwasser, "A survey of buffer management policies for packet switches," ACM SIGACT News, Vol.41, No. 1, pp.100–128, 2010.

[19] E. Hahne, A. Kesselman and Y. Mansour, "Competitive buffer management for shared-memory switches," *In Proc. of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 53–58, 2001.

[20] T. Itoh and T. Nagumo, "Improved lower bounds for competitive ratio of multi-queue switches in QoS networks," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E88-A, No. 5, pp. 1155–1165, 2005.

[21] T. Itoh and N. Takahashi, "Competitive analysis of multi-queue preemptive QoS algorithms for general priorities," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E89-A, No. 5, pp. 1186–1197, 2006.

[22] J. Kawahara, K. M. Kobayashi, and T. Maeda, "Tight analysis of priority queuing policy for egress traffic," *Computer Networks*, Vol. 91, No. 4, pp. 614–624, 2015.

[23] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko, "Buffer overflow management in QoS switches," *SIAM Journal on Computing*, Vol. 33, No. 3, pp. 563–583, 2004.

[24] A. Kesselman and Y. Mansour, "Harmonic buffer management policy for shared memory switches," *Theoretical Computer Science*, Vol. 324, No. 2-3, pp. 161–182, 2004.

[25] A. Kesselman, Y. Mansour and R. van Stee, "Improved competitive guarantees for QoS buffering," *Algorithmica*, Vol.43, No.1-2, pp. 63–80, 2005.

[26] A. Kesselman and A. Rosén, "Scheduling policies for CIOQ switches," *Journal of Algorithms*, Vol. 60, No. 1, pp. 60–83, 2006.

[27] A. Kesselman and A. Rosén, "Controlling CIOQ switches with priority queuing and in multistage interconnection networks," *Journal of Interconnection Networks*, Vol. 9, No. 1/2, pp. 53–72, 2008.

[28] A. Kesselman, K. Kogan and M. Segal, "Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing," *Distributed Computing*, Vol.23, No.3, pp. 163–175, 2010.

[29] A. Kesselman, K. Kogan and M. Segal, "Best effort and priority queuing policies for buffered crossbar switches," *Chicago Journal of Theoretical Science*, pp. 1–14, 2012.

[30] A. Kesselman, K. Kogan and M. Segal, "Improved competitive performance bounds for CIOQ switches," *Algorithmica*, Vol.63, No.1-2, pp, 411–424, 2012.

[31] K. Kobayashi, S. Miyazaki and Y. Okabe, "A tight bound on online buffer management for two-port shared-memory switches," *In Proc. of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 358–364, 2007.

[32] K. Kobayashi, S. Miyazaki and Y. Okabe, "A tight upper bound on online buffer management for multi-queue switches with bicodal buffers," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E91-D, No. 12, pp. 2757–2769, 2008.

[33] K. Kogan, A. Lopez-Ortiz, S. Nikolenko, and A. Sirotkin, "Multi-queued network processors for packets with heterogeneous processing requirements," *In Proc. of the 5th International Conference on Communication Systems and Networks*, pp. 1–10, 2013.

[34] Z. Lotker and B. Patt-Shamir, "Nearly optimal FIFO buffer management for two packet classes," *Computer Networks*, Vol. 42, No. 4, pp. 481–492, 2003.

[35] S. I. Nikolenko and K. Kogan, "Single and Multiple Buffer Processing," *In Encyclopedia of Algorithms*, pp. 1–9, *Springer*, 2015.

[36] M. Schmidt, "Packet buffering: Randomization beats deterministic algorithms," *In Proc. of the 22nd International Symposium on Theoretical Aspects of Computer Science*, pp. 293–304, 2005.

[37] M. Schmidt, "Online packet buffering," PhD thesis, Albert-Ludwigs-Universität Freiburg, 2006.

[38] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Communications of the ACM*, Vol. 28, No. 2, pp. 202–208, 1985.

[39] M. Sviridenko, "A lower bound for on-line algorithms in the FIFO model," unpublished manuscript, 2001.

[40] A. Zhu, "Analysis of queueing policies in QoS switches," *Journal of Algorithms*, Vol. 53, No. 2, pp. 137–168, 2004.