<table>
<thead>
<tr>
<th>Title</th>
<th>Competitive buffer management for multi-queue switches in QoS networks using packet buffering algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Kobayashi, Koji M.; Miyazaki, Shuichi; Okabe, Yasuo</td>
</tr>
<tr>
<td>Citation</td>
<td>Theoretical Computer Science (2017), 675: 27-42</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2017-05-02</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/226938">http://hdl.handle.net/2433/226938</a></td>
</tr>
<tr>
<td>Rights</td>
<td>© 2017. This manuscript version is made available under the CC-BY-NC-ND 4.0 license. The full-text file will be made open to the public on 2 May 2019 in accordance with publisher's 'Terms and Conditions for Self-Archiving'. This is not the published version. Please cite only the published version. この論文は出版社版ではありません。引用の際には出版社版をご確認ご利用ください。</td>
</tr>
<tr>
<td>Type</td>
<td>Journal Article</td>
</tr>
<tr>
<td>Textversion</td>
<td>author</td>
</tr>
<tr>
<td>Textversion</td>
<td><a href="http://creativecommons.org/licenses/by-nc-nd/4.0/">http://creativecommons.org/licenses/by-nc-nd/4.0/</a></td>
</tr>
</tbody>
</table>
Competitive Buffer Management for Multi-Queue Switches in QoS Networks Using Packet Buffering Algorithms

Koji M. Kobayashi\textsuperscript{1}, Shuichi Miyazaki\textsuperscript{2} and Yasuo Okabe\textsuperscript{2}
\textsuperscript{1}National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan
Corresponding author: kobaya@nii.ac.jp
\textsuperscript{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 $x = \min \{c + 2 - c, \alpha \}$, 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...
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 $\sigma$, an online policy $ALG$ obtains at least $1/c$-fraction of the value of an optimal offline policy for $\sigma$, 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 $ac$-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 $(c + \frac{2-c}{a(2-c)+c-1})$-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 $\min\{c + \frac{2-c}{a(2-c)+c-1}, ca\}$. 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 $\min\{c + \frac{2-c}{a(2-c)+c-1}, ca\}^2$.
Table 1: Competitive ratios for the 2-value multi-queue switch models

<table>
<thead>
<tr>
<th></th>
<th>Non-Preemptive</th>
<th>Preemptive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
<td>Upper Bound</td>
</tr>
<tr>
<td>Deterministic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>1 + \frac{1}{\alpha \ln(\alpha/(\alpha-1))} [20]</td>
<td>4 - \frac{2}{\alpha} * [9]</td>
</tr>
<tr>
<td></td>
<td>3.177* (C5.1)</td>
<td></td>
</tr>
<tr>
<td>Randomized</td>
<td>\frac{e}{e-1} \approx 1.581 [13]</td>
<td>\frac{e}{e-1} \approx 1.581 [13]</td>
</tr>
</tbody>
</table>

* large enough B, †any B, ‡B ≥ ln(α) + 2

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

<table>
<thead>
<tr>
<th></th>
<th>Non-Preemptive</th>
<th>Preemptive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
<td>Upper Bound</td>
</tr>
<tr>
<td>Deterministic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Randomized</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-value</td>
<td>ln(\alpha)/2 + 1 [40]</td>
<td>1.197 [5]</td>
</tr>
<tr>
<td>multi-value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* large enough B, †B ≥ 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 \(\alpha_i\) arrive at the \(i\)th 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 $e/e - 1 + o(1) (\approx 1.582)$ 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 $e/e - 1 (\approx 1.581)$ 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 $e/e - 1 (\approx 1.58)$-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 $e/e - 1$ 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 $\alpha$ is known to an algorithm in advance. We call a packet with value $1$ (respectively) a 1-packet (an $\alpha$-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 $\sigma$ is denoted by $V_{ALG}(\sigma)$. For an online algorithm $ALG$, if for any input $\sigma$, $V_{ALG}(\sigma) \geq V_{OPT}(\sigma)/c$, then we say that $ALG$ is $c$-competitive, where $OPT$ is an optimal offline algorithm for $\sigma$. 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 $j$th 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 $\alpha$-packets and the other for 1-packets. Therefore, it is convenient for us to think that $DS$ is equipped with two $A$s. To distinguish these two algorithms, we call them in different names, “AS” for scheduling of $\alpha$-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 a 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 \( Q^{(t)} \) at \( t \), (ii) OS stores \( B \) packets at \( Q^{(t)} \) 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 \( Q^{(t)} \) at \( t_- \).

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

### 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 \( \tau \) on \( \sigma_r \), if DS stores a 1-packet \( p \) at \( \tau \), then OS stores \( p' \) such that \( X(p') = p \) at \( \tau \).

**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 hold a packet. Let \( t \) be an event time on \( \sigma_r \). 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 $Q(i)$)

**DS’s execution:** $X(p) := p$. If $h^{(i)}_{DS}(t_{-}) < B$, DS accepts $p$. If $p$ is an $\alpha$-packet, $h^{(i)}_{DS}(t_{-}) = B$ and DS stores a 1-packet in $Q(i)$ at $t_{-}$, DS preempts it and accepts $p$. Otherwise, DS rejects $p$. If $p$ is an $\alpha$-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 $h^{(i)}_{OS}(t_{-}) = B$ and DS accepts $p$, DS sets $X(p') := p$ in which $p'$ is a packet such that $OS$ stores a packet $p'$ in $Q(i)$ at $t_{-}$ but DS does not store $X(p')$ in $Q(i)$ at $t_{-}$. Such $p'$ exists by Lemma 3.1.

**Case S (t is a scheduling event time):**

**Case AS (DS stores at least one $\alpha$-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 $\alpha$-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 $\sigma_{r}$ be an input for $DS$. Let $f_{AS}(\sigma_{r})$ be an input constructed by removing all the arrival events of 1-packets from $\sigma_{r}$. 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 $\alpha$-packets are passed to $AS$, and $AS$ accepts them greedily and may execute more than one scheduling events at one scheduling event of $\sigma_{r}$. It is then not hard to see that $AS$’s behavior on $\sigma_{r}$ 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 \( \sigma_r \) by removing all the scheduling events where \( DS \) executes Case AS. The construction of \( g_{OS}(\sigma_r) \) from \( f_{AS}(\sigma_r) \) 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, \( g_{OS}(\sigma_r) = (e_1, e_2, e_2', e_3, e_4, e_5, e_6, e_6') \), 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 \( \tau \) on \( \sigma_r \), \( \forall i \ h_{OS}^{(i)}(\tau) \geq h_{DS}^{(i)}(\tau) \).

**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 \( \sigma_r \). We assume that the statement is true at time \( t_- \) and show that it is true at \( t_+ \), namely, we assume that \( \forall i \ h_{OS}^{(i)}(t_-) \geq h_{DS}^{(i)}(t_-) \) and show that \( \forall i \ h_{OS}^{(i)}(t_+) \geq h_{DS}^{(i)}(t_+) \).

**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 \( \forall i (\neq j) h_{OS}^{(i)}(t_+) \geq h_{DS}^{(i)}(t_+) \). Thus, in what follows, we will show \( h_{OS}^{(j)}(t_+) \geq h_{DS}^{(j)}(t_+) \).

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

**Case 2: \( t \) is a scheduling event time and \( DS \) executes Case AS:** By definition, the number of arriving \( \alpha \)-packets in \( f_{AS}(\sigma_r) \) is equal to that in \( \sigma_r \), and moreover both \( DS \) and \( AS \) greedily accept \( \alpha \)-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 \( \alpha \)-packet at \( t \). Hence, \( h_{DS}^{(j)}(t_+) = h_{DS}^{(j)}(t_-) - 1 \), where \( DS \) transmits the \( \alpha \)-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, \( \forall i h_{OS}^{(i)}(t_+) = h_{OS}^{(i)}(t_-) \) 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,

\[
\forall i h_{OS}^{(i)}(t_+) = h_{OS}^{(i)}(t_-) - x_i - y_i.
\]

All the packets which \( DS \) stores at \( t_- \) are 1-packets because \( DS \) does not store \( \alpha \)-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.
Case 2: \( DS \) transmits a packet when \( OS \) returns one in Case OS,

\[
\forall i \ h_{OS}^{(i)}(t_-) = h_{DS}^{(i)}(t_-) + x_i.
\]

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 \( \sigma \) on \( M \) and an algorithm \( ALG \) for \( M \), define \( T_{ALG}(\tau) \) as the number of packets transmitted by \( ALG \) on \( \sigma \).

**Lemma 3.4** \( R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \geq T_{OS}(gos(\sigma_r)). \)

**Proof.** For a non-event time \( \tau \) on \( \sigma_r \), 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 \( \sigma_r \) before \( \tau \), and let \( T_{OS}(gos(\sigma_r), \tau) \) denote the number of packets transmitted by \( OS \) before \( \tau \). To prove this lemma, it suffices to prove the following inequality for any non-event time \( \tau \) on \( \sigma_r \):

\[
R_{AS}(\sigma_r, \tau) + R_{OS}(\sigma_r, \tau) + \sum_{i=1}^{m} h_{DS}^{(i)}(\tau) \geq T_{OS}(gos(\sigma_r), \tau) + \sum_{i=1}^{m} h_{OS}^{(i)}(\tau). \tag{1}
\]

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 \( T_{OS}(gos(\sigma_r), t_F) = T_{OS}(gos(\sigma_r)) \). By the above argument and Eq. (1), \( R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \geq T_{OS}(gos(\sigma_r)) \).

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 \( \sigma_r \). 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 \( T_{OS}(gos(\sigma_r), t_+) = T_{OS}(gos(\sigma_r), t_-) \). 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 \( \alpha \)-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, \( T_{OS}(gos(\sigma_r), t_+) = \)
\( T_{\text{OS}}(g_{\text{OS}}(\sigma_r), t_\text{+}) \), \( R_{\text{OS}}(\sigma_r, t_\text{+}) = R_{\text{OS}}(\sigma_r, t_-) \), and \( \sum_{i=1}^{m} h_{\text{OS}}^{(i)}(t_\text{+}) = \sum_{i=1}^{m} h_{\text{OS}}^{(i)}(t_-) \). 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 \( \alpha \)-packet at \( t_- \) by the condition of Case OS, \( R_{\text{AS}}(\sigma_r, t_\text{+}) = R_{\text{AS}}(\sigma_r, t_-) \). Let \( x_i \) be the number of packets \( p' \) such that \( OS \) transmits \( p' \) from \( Q_{\text{OS}}(\sigma_r) \) at \( t_- \). Moreover, let \( y_i \in \{0, 1\} \) be the number of 1-packets \( p'' \) such that \( DS \) 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,

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

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

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

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

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

and

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

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

## 4 Competitive Analysis of \( DS \)

In this section, we prove the following theorem:

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

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

### 4.1 Bounding the Numbers of \( \alpha \)-packets

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

**Lemma 4.2** \( T_{B,\alpha}(\sigma_r) = 0 \).
Proof. DS accepts arriving \( \alpha \)-packets greedily, namely, it accepts an arriving \( \alpha \)-packet whenever its destination buffer has at most \( B - 1 \) \( \alpha \)-packets. Since DS never preempts an \( \alpha \)-packet as we have discussed in Sec. 3.1, those \( \alpha \)-packets are eventually transmitted. Thus, Condition (ii) of the definition of \( T_{B,\alpha}(\sigma_r) \) fails and hence, \( T_{B,\alpha}(\sigma_r) = 0 \) holds.

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

**Lemma 4.3** For any input \( \sigma_r \) for \( M_r \), there exists an input \( \sigma'_r \) for \( M_r \) such that \( T_{B,1}(\sigma'_r) = 0 \) and \( \frac{V_{OPT_r}(\sigma'_r)}{V_{DS}(\sigma'_r)} \geq \frac{V_{OPT_r}(\sigma_r)}{V_{DS}(\sigma_r)} \).

**Proof.** If \( T_{B,1}(\sigma_r) = 0 \), then we are done (simply let \( \sigma'_r = \sigma_r \)). If \( T_{B,1}(\sigma_r) > 0 \), then let \( p_j \) \( (j = 1, \ldots, T_{B,1}(\sigma_r)) \) be a 1-packet satisfying the condition of \( T_{B,1}(\sigma_r) \). Then, construct an input \( \sigma'_r \) from \( \sigma_r \) by replacing each 1-packet \( p_j \) with an \( \alpha \)-packet \( q_j \). Since at each arrival of \( q_j \), DS’s corresponding buffer is full of \( \alpha \)-packets, DS rejects \( q_j \). Therefore, the behavior of DS is exactly the same for \( \sigma_r \) and \( \sigma'_r \), thus \( V_{DS}(\sigma'_r) = V_{DS}(\sigma_r) \). We can define an offline algorithm \( OFF \) for \( \sigma'_r \) whose behavior is exactly the same as that of \( OPT_r \) for \( \sigma_r \). Then since \( OFF \) transmits \( q_j \) for every \( j \), \( V_{OPT_r}(\sigma'_r) \geq V_{OFF}(\sigma'_r) \geq V_{OPT_r}(\sigma_r) + T_{B,1}(\sigma_r)(\alpha - 1) \geq V_{OPT_r}(\sigma_r) \). This completes the proof. \( \Box \)

By Lemmas 4.2 and 4.3, the numbers of 1-packets and \( \alpha \)-packets respectively that \( OPT_r \) transmits but DS does not are \( T_{B,1}(\sigma_r) \) and \( T_{B,\alpha}(\sigma_r) \). Let \( T_1(\sigma_r) \) \( (T_\alpha(\sigma_r)) \) denote the number of 1-packets \( (\alpha \)-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,

\[
V_{OPT_r}(\sigma_r) = V_{common}(\sigma_r) + T_{B,1}(\sigma_r) + \alpha T_{B,\alpha}(\sigma_r)
\]

and

\[
V_{DS}(\sigma_r) = V_{common}(\sigma_r) + T_1(\sigma_r) + \alpha T_\alpha(\sigma_r).
\]

By these equalities,

\[
V_{OPT_r}(\sigma_r) = V_{DS}(\sigma_r) - T_1(\sigma_r) - \alpha T_\alpha(\sigma_r) + T_{B,1}(\sigma_r) + \alpha T_{B,\alpha}(\sigma_r).
\]

(2)

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

\[
V_{DS}(\sigma_r) = \alpha R_{AS}(\sigma_r) + R_{OS}(\sigma_r).
\]

(3)

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

**Lemma 4.4**

\[
T_{B,\alpha}(\sigma_r) - T_\alpha(\sigma_r) \leq (c - 1)R_{AS}(\sigma_r).
\]

(4)

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

\[
T_{AS}(g_{AS}(\sigma_r)) = R_{AS}(\sigma_r)
\]

(5)

and

\[
g_{AS}(\sigma_r) = f_{AS}(\sigma_r).
\]

(6)
By the definitions of $c$ and $AS$ (i.e. $A$),
\[ cT_{AS}(f_{AS}(\sigma_r)) \geq T_{OPT_1}(f_{AS}(\sigma_r)). \] (7)

Let $OPT_{r,\alpha}$ be the offline algorithm which accepts and transmits only $\alpha$-packets transmitted by $OPT_r$. Using Lemma 4.2, we have
\[ T_{OPT_1}(f_{AS}(\sigma_r)) \geq T_{OPT_{r,\alpha}}(\sigma_r) = T_{B,\alpha}(\sigma_r) + T_{common} = T_{B,\alpha}(\sigma_r) + T_{common}, \] (8)

where $T_{common}$ denotes the number of $\alpha$-packets which both $OPT_r$ and $DS$ transmit, and the inequality follows from the optimality of $OPT_1$. Also, we have
\[ R_{AS}(\sigma_r) = T_{\alpha}(\sigma_r) + T_{common}. \] (9)

Then,
\[ cR_{AS}(\sigma_r) = cT_{AS}(g_{AS}(\sigma_r)) = cT_{AS}(f_{AS}(\sigma_r)) \quad \text{(by Eqs. (5) and (6))} \]
\[ \geq T_{OPT_1}(f_{AS}(\sigma_r)) \geq T_{B,\alpha}(\sigma_r) + T_{common} \quad \text{(by Eqs. (7) and (8))} \]
\[ = T_{B,\alpha}(\sigma_r) + R_{AS}(\sigma_r) - T_{\alpha}(\sigma_r). \quad \text{(by Eq. (9))} \]

The proof can be completed by rearranging this inequality. \hfill \square

### 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 $T_{B,1}(\sigma_r) - T_{1}(\sigma_r)$ from above. As we have seen in Lemma 4.4, bounding $T_{B,\alpha}(\sigma_r) - T_{\alpha}(\sigma_r)$ 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, $\bar{p}$-events, and $\bar{f}$-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 $\sigma$ is an input in $\{\sigma_r, g_{OS}(\sigma_r)\}$, and $OFF$ is an offline algorithm for $\sigma$ which greedily accepts arriving packets and can preempt a packet only when its buffer is full. At a non-event time $t$, suppose that $h_{OS}(i)(t) - h_{OFF}(i)(t) \geq 0$. Then, the number of $OFF$’s free cells in $Q(i)$ at $t$ is defined as $h_{OS}(i)(t) - h_{OFF}(i)(t)$. Similarly, if $h_{OFF}(i)(t) - h_{OS}(i)(t) \geq 0$, then the number of $OS$’s free cells in $Q(i)$ at $t$ is $h_{OFF}(i)(t) - h_{OS}(i)(t)$.

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 $(OFF, \sigma)$. We will omit “for $(OFF, \sigma)$” 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 $h_{OS}(i)(t) - h_{OFF}(i)(t) - 1$ just before $e_s$ and $h_{OS}(i)(t) - h_{OFF}(i)(t)$ just after $e_s$, and (iii) after $e_s$, the
number of $\text{OFF}$’s free cells in $Q^{(i)}$ retains at least $h_{\text{OS}}^{(i)}(t_{-}) - h_{\text{OFF}}^{(i)}(t_{-})$ 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 $(\text{OFF}, \sigma)$ if the corresponding $p$-event is for $(\text{OFF}, \sigma)$.

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

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

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

**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 $\mathcal{P}$-events is equal to that of $\mathcal{F}$-events. \(\square\)

### 4.3 Bounding the Numbers of 1-packets

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

**Lemma 4.6** $T_{B,\alpha}(\sigma_r) + \mathcal{F}_{\text{B,1}}(\sigma_r) - T_\alpha(\sigma_r) - T_1(\sigma_r) \leq \mathcal{F}_{\text{OPT}_r}(\sigma_r) - \mathcal{P}_{\text{OPT}_r}(\sigma_r)$.

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

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

$\mathcal{F}_\alpha(\sigma_r, t') \mathcal{T}_1(\sigma_r, t'))$ denotes the number of α-packets (1-packets) $p$ such that (i) $p$ arrives at a time $t'' < t'$, and (ii) $\text{DS}$ does not drop $p$ before $t'$ but $\text{OPT}_r$ drops $p$ before $t'$. 13
<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Action</th>
<th>OS</th>
<th>OFF</th>
<th>Free cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>scheduling</td>
<td>Only OS schedules</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>scheduling</td>
<td>Only OFF schedules</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>scheduling</td>
<td>Only OFF schedules</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>scheduling</td>
<td>Only OFF schedules</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>scheduling</td>
<td>Only OFF schedules</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>scheduling</td>
<td>Only OS schedules</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>scheduling</td>
<td>Only OFF schedules</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>arrival</td>
<td>A packet arrives</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>scheduling</td>
<td>Only OFF schedules</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>scheduling</td>
<td>OFF and OS schedule</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>scheduling</td>
<td>OFF and OS schedule</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>scheduling</td>
<td>Only OS schedules</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>scheduling</td>
<td>OFF and OS schedule</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Now we prove by induction on time that for a non-event time \( t' \) on \( \sigma_r \),

\[
T_{B,\alpha}(\sigma_r, t') + T_{\overline{B},1}(\sigma_r, t') - T_{\alpha}(\sigma_r, t') - T_1(\sigma_r, t') \leq p_{\text{OPT},\alpha}(\sigma_r, t') - \overline{p}_{\text{OPT},\alpha}(\sigma_r, t').
\]

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 \( \sigma_r \), 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: \( \text{OPT} \), accepts \( p \) without preemption:** An arrival event at which \( \text{OPT} \) accepts the arriving packet is not an \( \overline{p} \)-event by definition. Thus, \( \overline{p}_{\text{OPT},\alpha}(\sigma_r, t_+) = \overline{p}_{\text{OPT},\alpha}(\sigma_r, t_-) \).

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

**Case 1.2: \( \text{DS} \) rejects \( p \):** Since \( \text{DS} \) rejects \( p \), \( T_{\alpha}(\sigma_r, t_-) \) and \( T_1(\sigma_r, t_-) \) do not change. Of course, \( p \) is a 1-packet or an \( \alpha \)-packet, and \( T_{B,\alpha}(\sigma_r, t_-) + T_{\overline{B},1}(\sigma_r, t_-) = T_{B,\alpha}(\sigma_r, t_-) + T_{\overline{B},1}(\sigma_r, t_-) + 1 \) by definition. Also, by the condition of Case 1.2, \( \text{DS} \) rejects \( p \), which means \( h_{\text{DS}}^{(i)}(t_-) = B \). Hence, \( h_{\text{OS}}^{(i)}(t_-) \geq h_{\text{DS}}^{(i)}(t_-) = B \) by Lemma 3.3. Then, \( \text{OS} \) rejects \( p \). That is, \( p_{\text{OPT},\alpha}(\sigma_r, t_+) = p_{\text{OPT},\alpha}(\sigma_r, t_-) + 1 \). The above argument and the induction hypothesis show that Eq. (10) is true.

**Case 1.3: \( \text{DS} \) preempts a packet \( p' \) and accepts \( p \):** By the greediness of \( \text{DS} \), \( p \) and \( p' \) are an \( \alpha \)-packet and a 1-packet, respectively. Thus, \( T_{B,\alpha}(\sigma_r, t_-) = T_{B,\alpha}(\sigma_r, t_-) \) and \( T_{\alpha}(\sigma_r, t_-) = T_{\alpha}(\sigma_r, t_-) \). If \( \text{OPT} \) does not drop \( p' \) before \( t_+ \), i.e., \( p' \) exists in its buffer at \( t_+ \) or \( \text{OPT} \) transmits \( p' \) before \( t_+ \), then \( T_{\overline{B},1}(\sigma_r, t_-) = T_{\overline{B},1}(\sigma_r, t_-) + 1 \) and \( T_1(\sigma_r, t_-) = T_1(\sigma_r, t_-) \). If \( \text{OPT} \) drops \( p' \) before \( t_+ \), i.e., \( \text{OPT} \) rejects or preempts \( p' \) before \( t_+ \), then \( T_{\overline{B},1}(\sigma_r, t_-) = T_{\overline{B},1}(\sigma_r, t_-) \) and \( T_1(\sigma_r, t_-) = T_1(\sigma_r, t_-) - 1 \). That is, in either case \( T_{\overline{B},1}(\sigma_r, t_-) - T_1(\sigma_r, t_-) = T_{\overline{B},1}(\sigma_r, t_-) - T_1(\sigma_r, t_-) + 1 \) holds. Moreover, since \( \text{DS} \) preempts \( p' \), \( h_{\text{DS}}^{(i)}(t_-) = B \). Then, \( h_{\text{OS}}^{(i)}(t_-) \geq h_{\text{DS}}^{(i)}(t_-) = B \) by Lemma 3.3. Hence, \( \text{OS} \) rejects \( p \). Thus, \( p_{\text{OPT},\alpha}(\sigma_r, t_+) = p_{\text{OPT},\alpha}(\sigma_r, t_-) + 1 \). Therefore, Eq. (10) is true by the above argument and the induction hypothesis.

**Case 2: \( \text{OPT} \) rejects \( p \):** Since a \( p \)-event does not occur in this case, \( p_{\text{OPT},\alpha}(\sigma_r, t_+) = p_{\text{OPT},\alpha}(\sigma_r, t_-) \).

**Case 2.1: \( \text{DS} \) accepts \( p \) without preemption:** \( T_{B,\alpha}(\sigma_r, t_-) + T_{\overline{B},1}(\sigma_r, t_-) \) does not change because \( \text{DS} \) accepts \( p \) which \( \text{OPT} \) rejects. Also, \( T_{\alpha}(\sigma_r, t_-) = T_{\alpha}(\sigma_r, t_-) + T_1(\sigma_r, t_-) + 1 \). Moreover, \( p_{\text{OPT},\alpha}(\sigma_r, t_-) \geq p_{\text{OPT},\alpha}(\sigma_r, t_-) + 1 \) because \( \text{OS} \) may accept \( p \). Thus, the above argument together with the induction hypothesis shows that Eq. (10) is true.

**Case 2.2: \( \text{DS} \) rejects \( p \):** Since both \( \text{OPT} \) and \( \text{DS} \) reject \( p \), the left side of Eq. (10) does not change. On the other hand, we can show that \( \text{OS} \) rejects \( p \) in the same way as Case 1.2 which is stated above. Hence, \( p_{\text{OPT},\alpha}(\sigma_r, t_+) = p_{\text{OPT},\alpha}(\sigma_r, t_-) \). 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, \( T_\alpha(\sigma, t_+) = T_\alpha(\sigma, t_-) + 1 \), and \( T_{B,\alpha}(\sigma, t_+) = T_{B,\alpha}(\sigma, t_-) \). Also, we can show \( T_{B,1}(\sigma, t_+) = T_{B,1}(\sigma, t_-) + 1 \) 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, \( \overline{P}_{OPT}(\sigma, t_+) = \overline{P}_{OPT}(\sigma, t_-) \). By the above argument and the induction hypothesis, Eq. (10) is true.

**Case 3: OPT preempts a packet p′ and accepts p:** Since a p-event does not occur in this case by definition, \( P_{OPT}(\sigma, t_+) = P_{OPT}(\sigma, t_-) \).

**Case 3.1: DS accepts p without preemption:** Since p is an α-packet, \( T_{B,\alpha}(\sigma, t_-) \) and \( T_\alpha(\sigma, t_-) \) 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 \( T_{B,1}(\sigma, t_+) = T_{B,1}(\sigma, t_-) \) and \( T_1(\sigma, t_+) = T_1(\sigma, t_-) + 1 \). If DS drops \( p'' \) before \( t_+ \), then \( T_1(\sigma, t_+) = T_1(\sigma, t_-) \). In addition, we assume that \( T_{B,1}(\sigma) = 0 \) 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, \( T_{B,1}(\sigma, t_+) = T_{B,1}(\sigma, t_-) - 1 \). That is, in either case, \( T_{B,1}(\sigma, t_+) - T_1(\sigma, t_+) = T_{B,1}(\sigma, t_-) - T_1(\sigma, t_-) - 1 \). Furthermore, OS may accept p and hence \( \overline{P}_{OPT}(\sigma, t_+) \geq \overline{P}_{OPT}(\sigma, t_-) + 1 \). 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:** \( T_\alpha(\sigma, t_-) \) does not change because p is an α-packet. Also, \( T_{B,\alpha}(\sigma, t_+) = T_{B,\alpha}(\sigma, t_-) + 1 \). Since \( h_{OS}^{(i)}(t_-) \geq h_{DS}^{(i)}(t_-) = B \) by Lemma 3.3, OS rejects p and thus \( \overline{P}_{OPT}(\sigma, t_+) = \overline{P}_{OPT}(\sigma, t_-) \). In regard to \( p'' \) in the same way as Case 3.1, \( T_{B,1}(\sigma, t_+) - T_1(\sigma, t_+) = T_{B,1}(\sigma, t_-) - T_1(\sigma, t_-) - 1 \). 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 \( h_{OS}^{(i)}(t_-) \geq h_{DS}^{(i)}(t_-) = B \) by Lemma 3.3. Thus, \( \overline{P}_{OPT}(\sigma, t_+) = \overline{P}_{OPT}(\sigma, t_-) \). Since both DS and OPT accept p, which is an α-packet, \( T_{B,\alpha}(\sigma, t_-) \) and \( T_\alpha(\sigma, t_-) \) do not change. If \( p' = p'' \), \( T_{B,1}(\sigma, t_-) \) and \( T_1(\sigma, t_-) \) 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 drops \( p' \) before \( t_+ \) and DS does not drop \( p'' \) before \( t_+ \), then \( T_{B,1}(\sigma, t_+) = T_{B,1}(\sigma, t_-) \) and \( T_1(\sigma, t_+) = T_1(\sigma, t_-) \). 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.

\[ F_{OPT}(\sigma) - \overline{F}_{OPT}(\sigma) \leq R_{AS}(\sigma) + (c - 1)(R_{AS}(\sigma) + R_{OS}(\sigma)) \]  \hspace{1cm} (11)

and

\[ F_{OPT}(\sigma) - \overline{F}_{OPT}(\sigma) \leq R_{AS}(\sigma) + R_{OS}(\sigma). \] \hspace{1cm} (12)

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

\[ k = T_{OS}(g_{OS}(\sigma)) - R_{OS}(\sigma). \] \hspace{1cm} (13)
and

\[ k' = R_{AS}(\sigma_r). \] (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 \( \sigma_r \) 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 \( T_{A'}(g_{OS}(\sigma_r)) \) denote the number of packets which \( A' \) transmits but \( OS \) does not. Also, let \( T_{OS}(g_{OS}(\sigma_r)) \) 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

\[ T_{A'}(g_{OS}(\sigma_r)) - T_{OS}(g_{OS}(\sigma_r)) \leq (c - 1)T_{OS}(g_{OS}(\sigma_r)), \] (15)

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

Now we consider the relation between the value of \( T_{A'}(g_{OS}(\sigma_r)) - T_{OS}(g_{OS}(\sigma_r)) \) 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 \( p \)-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, \( T_{A'}(g_{OS}(\sigma_r)) = z_1, T_{OS}(g_{OS}(\sigma_r)) = w_1 + w_2 \), \( P_{A'}(g_{OS}(\sigma_r)) = \sum_{i=1}^{2} z_i \), \( \overline{P}_{A'}(g_{OS}(\sigma_r)) = \sum_{i=1}^{3} w_i \), and \( z_2 = w_3 \). By these equalities, we have

\[ P_{A'}(g_{OS}(\sigma_r)) - \overline{P}_{A'}(g_{OS}(\sigma_r)) = T_{A'}(g_{OS}(\sigma_r)) - T_{OS}(g_{OS}(\sigma_r)). \] (16)

By the definition of \( A', OPT_r \) for \( \sigma_r \), \( A' \) 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 \( (OPT_r, \sigma_r) \) as well by the definition of \( p \)-events. Also, if an arrival event is not an \( p \)-event for \( (A', g_{OS}(\sigma_r)) \), then it is not an \( p \)-event for \( (OPT_r, \sigma_r) \) either by the definition of \( p \)-events. Let \( x' \) denote the number of arrival events which are \( p \)-events for \( (OPT_r, \sigma_r) \) but not for \( (A', g_{OS}(\sigma_r)) \). Similarly, let \( y' \) denote the number of arrival events which are \( p \)-events for \( (A', g_{OS}(\sigma_r)) \) but not for \( (OPT_r, \sigma_r) \). Then, we have

\[ P_{OPT_r}(\sigma_r) = P_{A'}(g_{OS}(\sigma_r)) + x' \] (17)

and

\[ \overline{P}_{OPT_r}(\sigma_r) = \overline{P}_{A'}(g_{OS}(\sigma_r)) - y'. \] (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,

\[ x' + y' \leq k'. \] (19)
By the above argument,
\[
\mathcal{F}_{OPT_r}(\sigma_r) - \mathcal{F}_{OPT_r}(\sigma_r) = \mathcal{P}_{OPT_r}(\sigma_r) - \mathcal{P}_{OPT_r}(\sigma_r) \quad \text{(by Lemma 4.5)}
\]
\[
= \mathcal{P}(g_{OS}(\sigma_r)) - \mathcal{P}(g_{OS}(\sigma_r)) + x' + y' \quad \text{(by Eqs. (17) and (18))}
\]
\[
\leq \mathcal{P}(g_{OS}(\sigma_r)) - \mathcal{P}(g_{OS}(\sigma_r)) + k' \quad \text{(by Eq. (19))}
\]
\[
\leq \mathcal{T}(g_{OS}(\sigma_r)) - \mathcal{T}(g_{OS}(\sigma_r)) + k' \quad \text{(by Eq. (16))}
\]
\[
\leq (c - 1)T_{OS}(g_{OS}(\sigma_r)) + R_{AS}(\sigma_r) \quad \text{(by Eqs. (15) and (14))}
\]
\[
\leq (c - 1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) + R_{AS}(\sigma_r) \quad \text{(by Lemma 3.4)}
\]

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 \((OPT_r, \sigma_r)\), \(OPT_r\) 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 \(k'\) \(OPT\)-events. Since only \(OS\) transmits packets at the \(k\) \(OS\)-events, they are not \(f\)-events. On the other hand, the \(k'\) \(OPT\)-events can be \(f\)-events. Hence, the number of \(f\)-events for \((OPT_r, \sigma_r)\) is at most \(T_{OS}(g_{OS}(\sigma_r)) + k' - k\). Therefore,

\[
\mathcal{F}_{OPT_r}(\sigma_r) - \mathcal{F}_{OPT_r}(\sigma_r) \leq \mathcal{F}_{OPT_r}(\sigma_r) \quad \text{(by the fact \(\mathcal{F}_{OPT_r}(\sigma_r) \geq 0\))}
\]
\[
\leq T_{OS}(g_{OS}(\sigma_r)) + k' - k
\]
\[
\leq T_{OS}(g_{OS}(\sigma_r)) + R_{AS}(\sigma_r) - T_{OS}(g_{OS}(\sigma_r)) + R_{OS}(\sigma_r) \quad \text{(by Eqs. (13) and (14))}
\]
\[
= R_{AS}(\sigma_r) + R_{OS}(\sigma_r).
\]

\[\square\]

**Lemma 4.8**

\[
T_{B,\alpha}(\sigma_r) + T_{B,1}(\sigma_r) - \mathcal{T}(\sigma_r) - \mathcal{T}(\sigma_r) \leq R_{AS}(\sigma_r) + (c - 1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)) \quad (20)
\]

and

\[
T_{B,\alpha}(\sigma_r) + T_{B,1}(\sigma_r) - \mathcal{T}(\sigma_r) - \mathcal{T}(\sigma_r) \leq R_{AS}(\sigma_r) + R_{OS}(\sigma_r). \quad (21)
\]

**Proof.** Using Lemmas 4.6 and 4.5, we have

\[
T_{B,\alpha}(\sigma_r) + T_{B,1}(\sigma_r) - \mathcal{T}(\sigma_r) - \mathcal{T}(\sigma_r) \leq \mathcal{P}_{OPT_r}(\sigma_r) - \mathcal{P}_{OPT_r}(\sigma_r)
\]
\[
\leq \mathcal{F}_{OPT_r}(\sigma_r) - \mathcal{F}_{OPT_r}(\sigma_r).
\]

This inequality and Lemma 4.7 directly imply Eqs. (20) and (21). \[\square\]

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

\[
R_{OS}(\sigma_r) \leq \frac{c - 1}{2 - c} R_{AS}(\sigma_r). \quad (22)
\]

Otherwise, we have

\[
R_{AS}(\sigma_r) < \frac{2 - c}{c - 1} R_{OS}(\sigma_r). \quad (23)
\]

18
Hence,

\[ V_{OPT}(\sigma_r) = V_{DS}(\sigma_r) - \frac{\alpha}{1 + \alpha} T_1(\sigma_r) - \alpha T_2(\sigma_r) + \alpha T_{B,\alpha}(\sigma_r) + \alpha T_{B,\alpha}(\sigma_r) \quad (\text{by Eq. (2)}) \]

\[ \leq V_{DS}(\sigma_r) + R_{AS}(\sigma_r) + R_{OS}(\sigma_r) + (\alpha - 1)(T_{B,\alpha}(\sigma_r) - T_{\alpha}(\sigma_r)) \quad (\text{by Eq. (21)}) \]

\[ \leq V_{DS}(\sigma_r) + R_{AS}(\sigma_r) + R_{OS}(\sigma_r) + (\alpha - 1)(c - 1)R_{AS}(\sigma_r) \quad (\text{by Eq. (4)}) \]

\[ = V_{DS}(\sigma_r) + \frac{(-\alpha + 2 + \alpha - c)R_{AS}(\sigma_r) + R_{OS}(\sigma_r)}{\alpha R_{AS}(\sigma_r) + R_{OS}(\sigma_r)} V_{DS}(\sigma_r). \quad (\text{by Eq. (3)}) \quad (24) \]

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

\[ V_{OPT}(\sigma_r) \leq V_{DS}(\sigma_r) + \frac{(-\alpha + 2 + \alpha - c)R_{AS}(\sigma_r) + \frac{\alpha - 1}{\alpha} R_{AS}(\sigma_r)}{\alpha R_{AS}(\sigma_r) + \frac{\alpha - 1}{\alpha} R_{AS}(\sigma_r)} V_{DS}(\sigma_r) \]

\[ = \left( c + \frac{2 - c}{\alpha(2 - c) + c - 1} \right) V_{DS}(\sigma_r). \]

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

\[ V_{OPT}(\sigma_r) \leq V_{DS}(\sigma_r) + \frac{(-\alpha + 2 + \alpha - c)\frac{2 - c}{\alpha(2 - c) + c - 1} R_{OS}(\sigma_r) + R_{OS}(\sigma_r)}{\alpha R_{AS}(\sigma_r) + R_{OS}(\sigma_r)} V_{DS}(\sigma_r) \]

\[ = \left( c + \frac{2 - c}{\alpha(2 - c) + c - 1} \right) V_{DS}(\sigma_r). \]

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 \(\alpha\) maximizing the competitive ratio; to calculate the competitive ratios and the corresponding values of \(\alpha\), 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 \(\frac{c}{2} \leq 1\) achieved by \(E^{MS\epsilon p^{*}}\) [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
most \min\{c + \frac{2-c}{\alpha(2-c)+c-1}, \alpha\}(2 - \frac{1}{\alpha})(\Delta d)$. Since an upper bound $c$ on the competitive ratio of $E^{M_{Ep}'}$ is $\frac{e}{c-1}$ 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 $-1-\alpha+2\alpha^2+\sqrt{1+2\alpha-3\alpha^2+4\alpha^3} (\Delta d')$-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 $\min\{c + \frac{2-c}{\alpha(2-c)+c-1}, \alpha\}d' (\Delta d)$. For each $B$, $d$ takes the maximum value provided in the fourth column of Table 3 by $\alpha$ 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 $E^{M_{Ep}'}$ 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 $E^{M_{Ep}'}$ is $\frac{e}{c-1}$ for large enough $B$. Therefore, the competitive ratio of a deterministic algorithm for the preemptive $M_2$ is at most $\min\{c + \frac{2-c}{\alpha(2-c)+c-1}, \alpha\}1.282$ where $c = \frac{e}{c-1}$, which takes the maximum value of $\frac{e}{c-1} \cdot \frac{3e-e^2-\sqrt{-16e+33e^2-22e^3+5e^4}}{4e-2e^2} \cdot 1.282 < 2.516$ when $\alpha = \frac{3e-e^2-\sqrt{-16e+33e^2-22e^3+5e^4}}{4e-2e^2}$.

---

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

<table>
<thead>
<tr>
<th>$B$</th>
<th>$c$</th>
<th>$\alpha$</th>
<th>Competitive Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>13/7</td>
<td>3.335</td>
<td>2.5596</td>
</tr>
<tr>
<td>3</td>
<td>35/18</td>
<td>3.647</td>
<td>2.5996</td>
</tr>
<tr>
<td>$\geq 4$ and even</td>
<td>17/9</td>
<td>3.453</td>
<td>2.5766</td>
</tr>
<tr>
<td>5</td>
<td>52/27</td>
<td>3.587</td>
<td>2.5931</td>
</tr>
<tr>
<td>7</td>
<td>23/12</td>
<td>3.554</td>
<td>2.5894</td>
</tr>
<tr>
<td>large enough</td>
<td>17/9</td>
<td>3.453</td>
<td>2.5766</td>
</tr>
</tbody>
</table>

The third column shows the values of $\alpha$ that maximize the competitive ratios.
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 $(1 + \frac{1}{2} - \alpha^{-1})$-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 $E^{M_{EP}}$ is $\frac{c}{c-1}$ for large enough $B$. Therefore, the competitive ratio of a randomized algorithm for the preemptive $M_2$ is at most $\min\{c + \frac{2 - c}{\alpha(2 - c) + c - 1}, \alpha\}(1 + \frac{1}{2} - \alpha^{-1})$ where $c = \frac{\epsilon}{c-1}$, which takes the maximum value of 2.2690 when $\alpha = 2.2454$. \qed

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 $1 \leq B \leq 8$ 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 $\min\{c + \frac{2 - c}{\alpha(2 - c) + c - 1}, \alpha\}(2 - \frac{1}{\alpha})(\triangleq d)$. 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 $\alpha = (89 + 2\sqrt{2314})/50$. \qed

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 $\frac{-1}{2} + 2\alpha^2 + \sqrt{1 + 2\alpha^2 - 3\alpha^2 + 4\alpha^2}(\triangleq d')$-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 $\min\{c + \frac{2 - c}{\alpha(2 - c) + c - 1}, \alpha\}d'(\triangleq d)$. For each value of $B$, $d$ takes the maximum value provided in the fourth column of Table 4 by $\alpha$ in the third column. \qed

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 $(1 + \frac{1}{2} - \alpha^{-1})$-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 $\min\{c + \frac{2 - c}{\alpha(2 - c) + c - 1}, \alpha\}(1 + \frac{1}{2} - \alpha^{-1})$ where $c = 16/13$, which takes the maximum value of $4(-13 + \sqrt{494} + 2\sqrt{5(7 + \sqrt{494})})/65 < 2.056$ when $\alpha = (7 + \sqrt{494})/20$. \qed
Table 4: Competitive ratios of deterministic online algorithms for the preemptive $M_2$ when $m = 2$

<table>
<thead>
<tr>
<th>$B$</th>
<th>$c$</th>
<th>$\alpha$</th>
<th>Competitive Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3/2</td>
<td>2.253</td>
<td>2.297</td>
</tr>
<tr>
<td>2, 3</td>
<td>4/3</td>
<td>1.853</td>
<td>2.169</td>
</tr>
<tr>
<td>4</td>
<td>13/10</td>
<td>1.776</td>
<td>2.145</td>
</tr>
<tr>
<td>5, 7</td>
<td>14/11</td>
<td>1.713</td>
<td>2.126</td>
</tr>
<tr>
<td>6</td>
<td>24/19</td>
<td>1.691</td>
<td>2.120</td>
</tr>
<tr>
<td>8</td>
<td>29/23</td>
<td>1.685</td>
<td>2.118</td>
</tr>
<tr>
<td>large enough</td>
<td>16/13</td>
<td>1.616</td>
<td>2.098</td>
</tr>
</tbody>
</table>

The third column shows the values of $\alpha$ 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 \( \min \{ c + \frac{2-c}{\alpha(2-c)+c-\alpha}, \alpha \} \), 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 $\alpha$-packets (than 1-packets), which we think is a correct direction in our policy. However, we use greedy algorithms within $\alpha$-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


