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

Koji Kobayashi
Graduate School of Informatics, Kyoto University
kobaya@net.ist.i.kyoto-u.ac.jp

Shuichi Miyazaki
Academic Center for Computing and Media Studies, Kyoto University
shuichi@media.kyoto-u.ac.jp

Yasuo Okabe
Academic Center for Computing and Media Studies, Kyoto University
okabe@media.kyoto-u.ac.jp

ABSTRACT

The online buffer management problem formulates the problem of queuing policies of network switches supporting QoS (Quality of Service) guarantee. We focus on multi-queue switches in QoS networks proposed by Azar et al. To achieve good upper bounds, they introduced so-called "the relaxed model". Also, they showed that if the competitive ratio of the single-queue model is at most c, and if the competitive ratio of the relaxed model is at most c', then the competitive ratio of the multi-queue switch model is cc'. They proved that $c' \leq 2$, and obtained upper bounds on the competitive ratios for several multi-queue switch models.

In this paper, we propose an online algorithm called DS (Dual Scheduling) for the competitive ratio of the relaxed model and obtain some better competitive ratios of the 2-value multi-queue switch model, where the value of packets is restricted to 1 and $\alpha(\geq 1)$. DS uses as subroutine any online algorithms A for the non-preemptive unit-value switch model, which has also been extensively studied. We prove that if the competitive ratio of A is at most c, then the competitive ratio of DS is at most $\frac{\alpha c(2-c)+c^2-2c+2}{\alpha(2-c)+c-1}$, which is strictly better than 2.

The followings are a couple of examples of the improvement on the competitive ratios of the 2-value multi-queue switch models using our result: (i) We have improved the competitive ratio of deterministic algorithms for the non-preemptive 2-value multi-queue switch model from 4 to 3.177 for large enough B, where B is the number of packets each queue can simultaneously store. (ii) We have proved that the competitive ratio of randomized algorithms for the non-preemptive 2-value multi-queue switch model is at most $\frac{17}{2} - \sqrt{30} \simeq 3.023$ for large enough B.

SPAA'09, August 11–13, 2009, Calgary, Alberta, Canada.

Categories and Subject Descriptors

C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design—*Packet-switching networks*; F.1.2 [**COMPUTATION BY ABSTRACT DEVICES**]: Modes of Computation—*Online computation*; F.2.2 [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*

General Terms

Algorithms, Performance, Theory

Keywords

Competitive analysis, Multi-Queue switches, Buffer management

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), where 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 transmission of packets using priority values. The goal of the buffer management algorithm is to maximize the total value of transmitted packets.

Recently, this kind of problem is modeled as online problems, and a great amount of work has been done. There have been proposed a lot of models, and the most basic one is the following [1]: A switch has a 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, namely, 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, to accept it or not. 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

by the competitive analysis [10, 26]. If, for any input σ, an online policy A obtains value at least $1/c$ of the optimal offline policy for σ, then we say that A is c-competitive.

Up to the present, several models have been considered. Among them, Azar et al. have introduced the *multi-queue switches model* [7]. 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 buffer (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 (namely, drop packets already in the buffer to make space) and accept the new packet. (We consider both models in which preemption is allowed and not.) At a scheduling event, an online policy selects 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 [2, 6, 7, 8, 13, 14, 25]. Table 1 summarizes the current best upper and lower bound results for several models. In the multi-value multi-queue switch model, $\alpha(\geq 1)$ is the ratio between the largest and the smallest values of packets. Among them, let us briefly review the technique in [7], which we improve in this paper.

In [7], the authors proposed a technique to convert an online algorithm for the single queue model into that of 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 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 c, and (ii) the competitive ratio of the preemptive relaxed model is at most c', then the competitive ratio of the corresponding multi-queue switch model is at most cc'. 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 present a new interesting technique to construct some algorithms for the 2-value multi-queue switch model, where the value of packets is restricted to 1 and $\alpha(\geq 1)$, using algorithms for the unit-value multi-queue switch model. As a result, we can show competitive ratios for the multi-value multi-queue switch model. In particular, we propose an algorithm $DS(A_1)$ for the preemptive relaxed model, where A_1 is an online algorithm for the unit-value multi-queue switch model that are used as subroutines. We prove that if the competitive ratio of A_1 is at most c, then the competitive ratio of $DS(A_1)$ is at most $\frac{\alpha c(2-c)+c^2-2c+2}{\alpha(2-c)+c-1}$. Using this result, we improve upper bounds on the competitive ratios of the 2-value multi-queue switch model, as summarized in Table 1 (Details are included in Sec. 5).

Note that Azar et al. [7] showed that improving competitive ratios for the single-queue models implies improving competitive ratios for the multi-queue switch models. 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. For the unit-value multi-queue model, a lot of works have been done. Azar et al. [7] gave a lower bound $1.366 - \Theta(1/m)$ of deterministic algorithms for any B, and an upper bound $\frac{e}{e-1}(\simeq 1.581)$ of a randomized algorithm. Albers et al. [2] 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 $17/9(\simeq 1.89)$-competitive deterministic algorithm for $B \geq 2$, and it is optimal in the case $B = 2$. Furthermore, a lower bound $\frac{e}{e-1}(\simeq 1.581)$ of online deterministic algorithms for any B and large enough m, and a lower bound 1.465 of online randomized algorithms for any B and large enough m were presented. Azar et al. [6] showed a $\frac{e}{e-1}(\simeq 1.58)$-competitive deterministic algorithm for $B > \log m$. Also, Schmidt [25] presented a 3/2-competitive randomized algorithm.

As for the single-queue models, the current upper and lower bounds on competitive ratios are summarized in Table 2.

The online buffer management for some switches such as shared-memory switches [12, 16, 22], CIOQ switches [18, 9, 21], and crossbar switches [19, 20] are extensively studied.

2. PRELIMINARIES

In this section, we formally define the problem studied in this paper, and the relaxed model introduced in [7].

2.1 Online Buffer Management Problem for Multi-Queue Switches

A multi-queue switch has m input ports (FIFO queues) each of which is equipped with a buffer whose size is 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 unit-value switch model, the value of any packet is identical, say one. In the 2-value switch model, which is studied in this paper, each packet takes one of two values, say, 1 and $\alpha(\geq 1)$. We call a packet with value 1 (α, respectively) a 1-packet (an α-packet, respectively).

An input is a sequence of events. An *event* is an *arrival event* or a *scheduling event*. At an arrival event, a packet (say, p) arrives at an input port (1 through m), and the task of an online algorithm (or an online policy) is to select one of the following actions: insert an arriving packet into the corresponding queue (*accept* p), drop it (*reject* p), or drop a packet p' existing in the current buffer and accept p (*preempt* p'). (We consider in this paper both preemptive and non-preemptive models.) If a packet is accepted, it is stored at the tail of the corresponding input queue. At a scheduling event, an online algorithm selects one nonempty input port from m ones and transmits the packet at the head of the selected queue.

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 A for an input σ is denoted by $V_A(\sigma)$. If $V_A(\sigma) \geq V_{OPT}(\sigma)/c$ for an arbitrary input σ, we say that A is *c-competitive*, where OPT is an optimal offline policy for σ. Without loss of generality, we can assume that OPT never preempts packets.

Table 1: Competitve ratios for the multi-queue switch models

		Non-Preemptive		Preemptive	
		Lower Bound	Upper Bound	Lower Bound	Upper Bound
deterministic algorithm	2-value	$1+\frac{1}{\alpha \ln(\alpha/(\alpha-1))}$ [13]	4 - $\frac{2}{\alpha}^{\dagger}$ [7]	$\frac{e}{e-1} \approx 1.58$ [2]	2.564^{*}, $2.6^{\dagger}$ [7]
			$3.177^{\ddagger}$ [this paper]		$2.465^{\ddagger}$ [this paper]
			$3.778^{\S}$ [this paper]		$2.577^{\S}$ [this paper]
	multi-value	$\ln(\alpha)+1$ [5]	$2\ln(\alpha)+4$ [7]	$\frac{e}{e-1} \approx 1.58$ [2]	$3-1/\alpha$ [14]
randomized algorithm	2-value	1.465 [2]		1.465 [2]	$2.5^{\ddagger}$ [7]
			$3.023^{\dagger}$ [this paper]		$2.214^{\ddagger}$ [this paper]
					$2.297^{\dagger}$ [this paper]
	multi-value	1.465 [2]		1.465 [2]	

$^{*}B \to \infty$, †any B, ‡large enough B, $^{\S}B \geq 2$

Table 2: Competitve ratios for the single-queue model

		Non-Preemptive		Preemptive	
		Lower Bound	Upper Bound	Lower Bound	Upper Bound
deterministic algorithm	2-value	$2-1/\alpha$ [1]	$2-1/\alpha$ [5]	1.28 [15, 27]	1.282 [11]
	multi-value	$\ln(\alpha)+1$ [5]	$\ln(\alpha)+2$ [4]	1.419 [17]	1.732 [11]
randomized algorithm	2-value			1.197 [3]	$1+\alpha^{-\frac{1}{2}}-\alpha^{-1}$ [3]
	multi-value				

2.2 The Relaxed Model

The relaxed model is the same as the usual preemptive 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 an input queue can be transmitted at a scheduling event, but in the relaxed model, any packet can be transmitted (namely, the buffer is not a queue). As is the case with the multi-queue switch model, we can assume, without loss of generality, that OPT never preempts. Throughout this paper, for simplicity, the 2-value multi-queue switch model (the unit-value multi-queue switch model and the preemptive relaxed model, respectively) is denoted by M_2 (M_1, and M_r, respectively). In addition, we denote OPT_2 (OPT_1 and OPT_r, respectively) optimal offline algorithms for M_2 (M_1 and M_r, respectively).

3. ALGORITHM DS

We propose Dual Scheduling Algorithm(DS) for M_r in Sec. 3.1, and analyze the competitive ratio of DS in Sec. 3.2.

3.1 Dual Scheduling Algorithm(DS)

In this section, we give the definition of Dual Scheduling Algorithm (DS). Let A_1 be a *work-conserving* online algorithm for M_1. An algorithm which transmits a packet at a scheduling event whenever its buffer is not empty is called work-conserving. (See [7], e.g.) DS uses A_1 as a subroutine, and hence it is written as $DS(A_1)$, but for simplicity, we write "DS" instead of "$DS(A_1)$" when A_1 is clear.

We give some definitions. For a time t when an event occurs, $t-$ represents a moment before t and after the previous event occurred. Similarly, $t+$ is a moment after t and before the next event occurs. The jth queue of the switch is denoted as $Q^{(j)} (1 \leq j \leq m)$. For an algorithm A for M_r or M_1, $h_A^{(j)}(t)$ denotes the number of packets A holds in $Q^{(j)}$ at time t when no event happens. For an algorithm A_r for M_r, $g_{A_r}^{(j)}(t)$ denotes the number of α-packets A_r holds in $Q^{(j)}$ at time t when an event does not happen. Let $\sigma(t)$ denote the prefix of the input σ up to time t. To define an algorithm, we need to specify its buffer management policy at an arrival event, and a scheduling policy at a scheduling event.

First, we sketch an outline of DS. When an arrival event happens, DS greedily accepts an α-packet. When a scheduling event occurs at time t, $DS(A_1)$ uses two subroutines $AS(A_1)$ (standing for *α-packet Scheduling algorithm*) and $OS(A_1)$ (standing for 1-*packet Scheduling algorithm*). (Hence A_1 is actually a *subsubroutine* of DS.) For simplicity, we write AS and OS instead of $AS(A_1)$ and $OS(A_1)$, respectively, when A_1 is clear.

DS calls AS if DS holds at least one α-packet. AS returns to DS the name of the α-packet which DS should transmit at t. Otherwise, DS calls OS, which returns to DS the name of the 1-packet which should be transmitted at t. (In M_r, it suffices to decide the *queue* from which a packet is transmitted. Instead we require AS and OS to decide the name of the *packet* to be transmitted for later analysis.) Specifically, AS constructs $\sigma'(t)$ by removing all arrival events where a 1-packet arrives within $[0, t]$ from $\sigma(t)$. Then, $AS(A_1)$ calls **Scheduling Routine** as a subsubroutine, and returns the name of the packet which A_1 should return to DS. On the other hand, $OS(A_1)$ constructs $\sigma''(t)$ by removing all scheduling events where $OS(A_1)$ is not called

by DS from $\sigma(t)$, and calls **Scheduling Routine**. For using A_1 as a subroutine of AS or OS, in **Scheduling Routine**, we assume that A_1 gives a priority to each arriving packet, and transmits the packet with the highest priority at each scheduling event.

Dual Scheduling Algorithm (DS)

Buffer Management: DS accepts packets greedily, namely, when a 1-packet p arrives at $Q^{(i)}$ at time t, DS accepts p if $h_{DS}^{(i)}(t-) < B$. Otherwise, p is rejected. When an α-packet q arrives at $Q^{(i)}$ at t, if $h_{DS}^{(i)}(t-) < B$, then DS accepts q. If $h_{DS}^{(i)}(t-) = B$ and $g_{DS}^{(i)}(t-) < B$, DS preempts a 1-packet and accepts q. Otherwise, DS rejects q.

Scheduling:
At a scheduling event at time t, execute one of the following cases:
Case D1: ($\sum_{i=1}^{m} g_{DS}^{(i)}(t-) > 0$, namely, DS has at least one α-packet):
DS calls AS, decides the α-packet p to be transmitted, and transmits p. DS finishes the execution at t.
Case D2: ($\sum_{i=1}^{m} g_{DS}^{(i)}(t-) = 0$, namely, DS does not have any α-packet):
DS calls OS, decides the 1-packet p to be transmitted, and transmits p. DS finishes the execution at t.

AS and OS are defined in the following:

α-Packet Scheduling Algorithm($AS(A_1)$):
(AS is called at time t.)
Step A1:
AS converts $\sigma(t)$ into $\sigma'(t)$ by removing all arrival events of 1-packets from $\sigma(t)$.
Step A2:
AS calls **Scheduling Routine**(AS, A_1, $\sigma'(t)$, t) (defined later), which returns the packet p. Then, AS returns p to DS, and finishes this routine at t.

1-Packet Scheduling Algorithm($OS(A_1)$):
(OS is called at time t.)
Step $\mathcal{O}$1:
OS converts $\sigma(t)$ into $\sigma''(t)$ by removing all scheduling events where OS is not called by DS before t.
Step $\mathcal{O}$2:
OS calls **Scheduling Routine**(OS, A_1, $\sigma''(t)$, t), which returns the packet p. Then, OS returns p to DS, and finishes this routine at t.

However, since two kinds of packets, namely 1-packets and α-packets can arrive at an arrival event in $\sigma''(t)$, A_1 cannot be run for $\sigma''(t)$ if nothing is done. So A_1 executes a buffer management for arriving packets according to the following definition.

Buffer Management for A_1: A_1 accepts 1-packets greedily, namely, when an α-packet q arrives at $Q^{(i)}$ at time t'', A_1 accepts q if $h_{A_1}^{(i)}(t''-) < B$. Otherwise, A_1 rejects q. When a 1-packet p arrives at $Q^{(i)}$ at t'', if $h_{A_1}^{(i)}(t''-) < B$, then A_1 accepts p. If $h_{A_1}^{(i)}(t''-) = B$ and there exists an α-packet q' in $Q^{(i)}$, A_1 preempts q' and accepts p. It is easy to see that if $h_{A_1}^{(i)}(t''-) = B$, there does not exist an α-packet q' in $Q^{(i)}$ and DS accepts, then A_1 preempts q'' which DS does not hold at $Q^{(i)}$ at $t-$, and accepts p. Otherwise, p is rejected. For simplicity, when A_1 accepts (rejects, respectively) a packet, we say that "OS accepts (rejects, respectively) it".

Scheduling Routine(ALG, A_1, δ, s_ℓ):
Step S0:
Let s_j denote the jth scheduling event within $[0, s_\ell]$. Let n_j be the number of arrival events within $[0, s_j]$ $(1 \le j \le \ell)$. Let x be the parameter, and $x := 1$.
Step S1:
Let y be the parameter, and $y := 1$.
A_1 prioritizes all arrival events which happen within $[0, s_x]$ according to its definition. (We need not consider the buffer management in the case $ALG = AS$ since each arriving packet in $\sigma'(t)$ is unique, namely, an α-packet. However, in the case $ALG = OS$, the buffer management for A_1 which is defined above must be used.) In this regard, a *marked* arrival event is certainly the lowest priority. (Marking is done in Step S2.)
Step S2:
Suppose that A_1 transmits the packet p which arrives at the yth highest priority arrival event. Then, p is *marked*. We say that *ALG transmits p*. If DS stores p at s_x- and $x < \ell$ holds, $x := x + 1$, and go back to Step S1. Else, if DS stores p at s_x- and $x = \ell$ holds, A_1 returns p, and this routine is finished at s_ℓ. Else, if DS does not store p at s_x- and $y < n_x$ holds, $y := y + 1$ and go back to Step S2. Else, if DS does not store p at s_x- and $y = n_x$ holds, this routine is finished at s_ℓ.

DS calls OS immediately after time τ when DS does not store any packet in its buffer. Note that OS does not return a packet by the definition of **Scheduling Routine**, but can transmit all 1-packets which OS stores but DS does not at τ. (This property is used in the proof of Lemma 3.10 and Corollary 3.11.)

Note that although OS simulates some buffer management algorithms as its subroutine, and OS itself is a subroutine of DS, OS can be viewed as an online algorithm for the unit-value buffer management problem. So, when working on an input sequence, we can consider buffers which OS uses, and hence naturally define $h_{OS}^{(i)}(t)$, the number of packets OS holds in $Q^{(i)}$ at time t.

3.2 Competitive Analysis of DS

3.2.1 Overview of the Analysis

For an input σ_r for M_r, 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) p arrives at $Q^{(i)}$ at time t where $g^{(i)}_{DS}(t-) = B$, (ii) DS drops p (namely, p is rejected at t since DS greedily accepts arriving packets), and (iii) OPT_r accepts p, which is eventually transmitted since OPT_r never preempts. For an input σ_r for M_r, let $\mathcal{T}_{\bar{B},1}(\sigma_r)$ ($\mathcal{T}_{\bar{B},\alpha}(\sigma_r)$, respectively) be the number of 1-packets (α-packets, respectively) p such that (i) p arrives at $Q^{(i)}$ at time t where $g^{(i)}_{DS}(t-) < B$, (ii) DS drops p, and (iii) OPT_r accepts p. Since DS accepts arriving packets greedily, if an α-packet is dropped from $Q^{(i)}$ at t, then $g^{(i)}_{DS}(t-) = B$. Therefore, $\mathcal{T}_{\bar{B},\alpha}(\sigma_r) = 0$ holds. We will prove in Lemma 3.2, that for any online algorithm A_r for M_r and for any input σ'_r for which the above defined $\mathcal{T}_{B,1}(\sigma'_r) > 0$, there exists another input σ''_r for which $\mathcal{T}_{B,1}(\sigma''_r) = 0$ and the competitive ratio of A_r is equal to or larger than that for σ'_r. Therefore, it suffices to consider only inputs σ_r for which $\mathcal{T}_{B,1}(\sigma_r) = 0$. Hence the numbers of 1-packets and α-packets, respectively, OPT_r accepts but DS drops are $\mathcal{T}_{\bar{B},1}(\sigma_r)$ and $\mathcal{T}_{B,\alpha}(\sigma_r)$. Then, $V_{OPT_r}(\sigma_r) \leq V_{DS}(\sigma_r) + \mathcal{T}_{\bar{B},1}(\sigma_r) + \alpha\mathcal{T}_{B,\alpha}(\sigma_r)$ holds. Let $R_A(\sigma_r)$ ($A = \{AS, OS\}$) be the number of packets returned by A for an input σ_r for M_r. Note that DS transmits a packet returned by AS or OS. Then $V_{DS}(\sigma_r) = R_{OS}(\sigma_r) + \alpha R_{AS}(\sigma_r)$ by definition. Let a D-event be a scheduling event where DS transmits an α-packet and OPT_r transmits a 1-packet. Let $\mathcal{K}$ be the number of D-events. Suppose that the competitive ratio of A_1, a subroutine of DS, is at most c (in M_1). In Sec. 3.2.2, we show that $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\} \geq \mathcal{T}_{B,\alpha}(\sigma_r)$, and in Sec. 3.2.3, we prove that $R_{AS}(\sigma_r) + \min\{(c-1)(R_{AS}(\sigma_r) + R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geq \mathcal{T}_{B,\alpha}(\sigma_r) + \mathcal{T}_{\bar{B},1}(\sigma_r)$. Therefore, $V_{OPT_r}(\sigma_r) \leq R_{OS}(\sigma_r) + \alpha R_{AS}(\sigma_r) + \mathcal{T}_{\bar{B},1}(\sigma_r) + \alpha\mathcal{T}_{B,\alpha}(\sigma_r) \leq \frac{\alpha c(2-c)+c^2-2c+2}{\alpha(2-c)+c-1} V_{DS}(\sigma_r)$. Hence, we have the following theorem:

THEOREM 3.1. *If the competitive ratio of A_1 for M_1 is at most c, then the competitive ratio of $DS(A_1)$ is at most $\frac{\alpha c(2-c)+c^2-2c+2}{\alpha(2-c)+c-1}$.*

3.2.2 Analysis of AS

Because of the space restriction, we omit the proofs of the following lemmas and corollaries. The complete proofs are included in the full version [23]. First, we show that it is sufficient to consider only inputs σ_r such that $\mathcal{T}_{B,1}(\sigma_r) = 0$.

LEMMA 3.2. *Let σ_r for M_r be an input for which $\mathcal{T}_{B,1}(\sigma_r) > 0$. Then, there exists an input σ'_r for M_r such that $\mathcal{T}_{B,1}(\sigma'_r) = 0$ and $\frac{V_{OPT_r}(\sigma'_r)}{V_{DS}(\sigma'_r)} \geq \frac{V_{OPT_r}(\sigma_r)}{V_{DS}(\sigma_r)}$.*

For analysis, we give some definitions. For an online algorithm A_1 for M_1, an input σ_1 for M_1, and a time t when no event occurs, we call the number of cells in a buffer such that A_1 holds a packet but OPT_1 does not hold a packet a *gap* for A_1, OPT_1 and σ_1. Namely, the gap at $Q^{(i)}$ at t for A_1, OPT_1 and σ_1 is $h^{(i)}_{A_1}(t) - h^{(i)}_{OPT_1}(t)$ if $h^{(i)}_{A_1}(t) - h^{(i)}_{OPT_1}(t) > 0$.

In what follows, we estimate the degree of increase and decrease of the gap at $Q^{(i)}$ at each event, namely, an arrival event and a scheduling event. Note that, at an arrival event in M_1, a strategy of greedily accepting arriving packets is the best way.

Arrival event:

A1. Let p be a packet which arrives at $Q^{(i)}$ at t:

A1.1. Both OPT_1 and A_1 accept p, namely, $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-) + 1$ and $h^{(i)}_{A_1}(t+) = h^{(i)}_{A_1}(t-) + 1$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-)$, the gap at $Q^{(i)}$ does not change at t.

A1.2. OPT_1 accepts p and A_1 rejects p, namely, $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-) + 1$ and $h^{(i)}_{A_1}(t+) = h^{(i)}_{A_1}(t-) = B$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-) - 1 \geq 0$, the gap at $Q^{(i)}$ decreases at t.

A1.3. OPT_1 rejects p, A_1 accepts or rejects p, namely, $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-) = B$, and $h^{(i)}_{A_1}(t+) \leq h^{(i)}_{A_1}(t-) + 1$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) \leq 0$, there does not exist the gap at $Q^{(i)}$ at $t-$ at $Q^{(i)}$, and the gap at $Q^{(i)}$ does not change at t .

A2. Let p be a packet which arrives at $Q^{(j)}$ ($j \neq i$) at t. $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-)$, and $h^{(i)}_{A_1}(t+) = h^{(i)}_{A_1}(t-)$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-)$, the gap at $Q^{(i)}$ at t does not change.

Scheduling event:

S1. Both OPT_1 and A_1 transmit from $Q^{(i)}$, namely, $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-) - 1$, and $h^{(i)}_{A_1}(t+) = h^{(i)}_{A_1}(t-) - 1$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-)$, the gap at $Q^{(i)}$ at t does not change.

S2. OPT_1 transmits from $Q^{(i)}$ and A_1 transmits from $Q^{(j)}$ ($j \neq i$), namely, $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-) - 1$, $h^{(i)}_{A_1}(t+) = h^{(i)}_{A_1}(t-)$:

S2.1. $h^{(i)}_{A_1}(t-) \geq h^{(i)}_{OPT_1}(t-)$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-) + 1 \geq 0$, the gap at $Q^{(i)}$ increases at t.

S2.2. $h^{(i)}_{A_1}(t-) < h^{(i)}_{OPT_1}(t-)$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-) + 1 < 1$, there does not exist the gap at $Q^{(i)}$ at $t-$ at $Q^{(i)}$, and the gap at $Q^{(i)}$ does not change at t.

S3. OPT_1 transmits from $Q^{(j)}$ ($j \neq i$), and A_1 transmits from $Q^{(i)}$, namely, $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-)$, $h^{(i)}_{A_1}(t+) = h^{(i)}_{A_1}(t-) - 1$:

S3.1. The gap at $Q^{(i)}$ is at least one at $t-$ namely, $h^{(i)}_{A_1}(t-) > h^{(i)}_{OPT_1}(t-)$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-) - 1$, the gap at $Q^{(i)}$ decreases at t.

S3.2. There does not exist the gap at $Q^{(i)}$ at $t-$, namely, $h^{(i)}_{A_1}(t-) \leq h^{(i)}_{OPT_1}(t-)$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-) + 1 \leq -1$, there does not exist the gap at $Q^{(i)}$ at $t-$ at $Q^{(i)}$, and the gap at $Q^{(i)}$ does not change at t.

S4. OPT_1 transmits from $Q^{(j)}$, and A_1 transmits from $Q^{(k)}$, namely, $h^{(i)}_{OPT_1}(t+) = h^{(i)}_{OPT_1}(t-)$, and $h^{(i)}_{A_1}(t+) = h^{(i)}_{A_1}(t-) - 1$:

Since $h^{(i)}_{A_1}(t+) - h^{(i)}_{OPT_1}(t+) = h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-)$, the gap at $Q^{(i)}$ does not change at t.

By the above estimation of the gap at $Q^{(i)}$, A_1 drops a packet from $Q^{(i)}$ at t if and only if the gap at $Q^{(i)}$ decreases in **A1.2.** Then, we call this arrival event a *p-event* (profit-event) for A_1, OPT_1 and an input σ_1 at M_1 at t. So, since $h^{(i)}_{OPT_1}(0) = h^{(i)}_{OPT_1}(0) = 0$ holds, the event satisfying **S2.1.**, namely, increasing the gap, certainly happens at $Q^{(i)}$ at $t'(< t)$ if an event satisfying **A1.2.** happens at

$Q^{(i)}$ at t. Then, for a p-event for A_1, OPT_1 and an input σ_1 at $Q^{(i)}$ at time t, the corresponding *g-event* (gap-event) for A_1, OPT_1 and an input σ_1 is a scheduling event that happens at $Q^{(i)}$ at t' satisfying the following three inequalities: $h^{(i)}_{A_1}(t'') - h^{(i)}_{OPT_1}(t'') \geq h^{(i)}_{A_1}(t-) - h^{(i)}_{OPT_1}(t-) = B - h^{(i)}_{OPT_1}(t-) \quad (\forall t'' \in [t'+, t-])$, $h^{(i)}_{A_1}(t'-) = h^{(i)}_{A_1}(t'+)$, and $h^{(i)}_{OPT_1}(t'-) = h^{(i)}_{OPT_1}(t'+) + 1$.

Here, we explain a p-event and a g-event using an example in Fig. 1. We consider the case $B = 4$. The input and action by OPT_1 and an online algorithm ON for M_1 are described in the column denoted by "event". Each column denoted by "OPT" and "ON" represent the number of packets OPT_1 and ON hold in $Q_{(i)}$, respectively. The rightmost column represents the gap at each timeslot. The right figure shows p-events and corresponding g-events.

For example, at each time 1,2,3 and 4, a packet arrives at $Q^{(i)}$, and both OPT_1 and ON accept it. Then, at time 5, a scheduling event happens and only ON transmits a packet from $Q^{(i)}$. On the other hand, at time 6 and 7, only OPT_1 transmits a packet from $Q^{(i)}$. An arrival event at time 16 is a p-event for OPT_1 and ON, and an event at time 7 is the corresponding g-event.

An online algorithm A_1, an optimal offline algorithm OPT_1 and an input σ_1 for σ_r decide whether an event is a p-event (g-event, respectively) or not. Hence, if an event e is a p-event (g-event, respectively), we write e is a p-event for (A_1, OPT_1, σ_1) (g-event for (A_1, OPT_1, σ_1), respectively). We may omit a 3-tuple (A_1, OPT_1, σ_1) when it is clear.

Here, we give some definitions on the number of p-events and g-events. For an input σ_1 at M_1, an optimal algorithm OPT_1 and an online algorithm A_1 for M_1, let $\mathcal{P}_{A_1}(\sigma_1)$ ($\mathcal{G}_{A_1}(\sigma_1)$, respectively) denote the number of p-events for (A_1, OPT_1, σ_1) (g-events for (A_1, OPT_1, σ_1), respectively). (Note that OPT_1 is not important for $\mathcal{P}_{A_1}(\sigma_1)$ and $\mathcal{G}_{A_1}(\sigma_1)$ since a value of $\mathcal{P}_{A_1}(\sigma_1)$ does not change due to an optimal offline algorithm.) Note that AS and OS can be regarded as A_1 since they convert an input σ_r for M_r into σ_1 for M_1, and decide a packet to be transmitted by DS.

LEMMA 3.3. *Let A_1 be an online algorithm for M_1 and σ_1 be an input for M_1. Then, $\mathcal{P}_{A_1}(\sigma_1) = \mathcal{G}_{A_1}(\sigma_1)$.*

Here, we give some definitions. For an input σ_r for M_r and a time t when no event happens, let $\mathcal{T}_{B,\alpha}(\sigma_r, t)$ be the number of α-packets p such that (i) p arrives at $Q^{(i)}$ at time t' where $g^{(i)}_{DS}(t'-) = B$, (ii) DS drops p at $t''(\in [t', t])$, and (iii) OPT_r accepts p at t'. For an input σ_1 for M_1, and an online algorithm A_1 for M_1, let $\mathcal{P}_{A_1}(\sigma_1, t)$ denote the number of p-events for (A_1, OPT_1, σ_1) which happen before t. Note that A_1 can be AS or OS, which can be regarded as an online algorithm for M_1. For any model M_1, or M_r, let σ be an input and A be an algorithm. Then define $T_A(\sigma)$ to be the number of transmitted packets by A for an input σ. When A_1 is simulated on $\sigma(t)$ where an event happens at time t by $OS(A_1)$, A_1 transmits some packets. Then, for better understanding, we say that OS transmits packets.

As we have done previously for M_1, we define a gap for AS and OS. For an input σ_r for M_r, we call the number of cells in a buffer such that AS holds a packet but OPT_r does not hold a packet a *gap* for AS, OPT_r and σ_r. Notice that the number of packets AS stores at time t is equal to that of α-packets DS holds at t. A gap for OS is defined similarly. Namely, the gap at $Q^{(i)}$ at t for AS, OPT_r and σ_r is $g^{(i)}_{DS}(t) - h^{(i)}_{OPT_r}(t)$ if $g^{(i)}_{DS}(t) - h^{(i)}_{OPT_r}(t) > 0$. Then, we call an arrival event where DS drops an α-packet from $Q^{(i)}$ a *p-event* for AS, OPT_r and σ_r at $Q^{(i)}$ at t. (We call an arrival event where OS drops a packet from $Q^{(i)}$ a *p-event* for OS, OPT_r and σ_r at $Q^{(i)}$ at t, respectively.) Also, for a p-event at $Q^{(i)}$ at time t, the corresponding *g-event* for AS, OPT_r and σ_r is a scheduling event that happens at t' satisfying the following three conditions: $g^{(i)}_{DS}(t'') - h^{(i)}_{OPT_r}(t'') \geq g^{(i)}_{DS}(t-) - h^{(i)}_{OPT_r}(t-) = B - h^{(i)}_{OPT_r}(t-) \quad (\forall t'' \in [t'+, t-])$, $g^{(i)}_{DS}(t'-) = g^{(i)}_{DS}(t'+)$, and $h^{(i)}_{OPT_r}(t'-) = h^{(i)}_{OPT_r}(t'+) + 1$. (For a p-event at $Q^{(i)}$ at time t, the corresponding *g-event* for OS, OPT_r and σ_r is a scheduling event that happens at t' satisfying the following three conditions:
$h^{(i)}_{OS}(t'') - h^{(i)}_{OPT_r}(t'') \geq h^{(i)}_{OS}(t-) - h^{(i)}_{OPT_r}(t-)$
$= B - h^{(i)}_{OPT_r}(t-)$ $(\forall t'' \in [t'+, t-])$, $h^{(i)}_{OS}(t'-) = h^{(i)}_{OS}(t'+)$, and $h^{(i)}_{OPT_r}(t'-) = h^{(i)}_{OPT_r}(t'+)+1$, respectively.) If an event e is a p-event (g-event, respectively), we write e is a p-event for (A, OPT_r, σ_r) where A is AS or OS (g-event for (A, OPT_r, σ_r), respectively). In this section, for an input σ_r for M_r, we show a relation between the number of α-packets which are not accepted by DS, namely $\mathcal{T}_{B,\alpha}(\sigma_r)$, and the number of p-events for (AS, OPT_r, σ_r), namely, $\mathcal{P}_{AS}(\sigma_r)$. In addition, for an input σ_r for M_r, we show an upper bound of the number on g-events for (AS, OPT_r, σ_r), namely, $\mathcal{G}_{AS}(\sigma_r)$.

LEMMA 3.4. *Let t be a time when no event happens, and σ_r be an input for M_r. Then, $\mathcal{P}_{AS}(\sigma_r, t) \geq \mathcal{T}_{B,\alpha}(\sigma_r, t)$.*

Recall that a D-event is a scheduling event at which OPT_r transmits a 1-packet and AS transmits an α-packet. In order to evaluate the number of g-events when $\mathcal{K}$ D-events happen, we consider a modification of M_1, which we call the *sleep model* (denoted by M_s). An input for M_s is a sequence of events. An event is an arrival event, an N-scheduling event (normal scheduling event) or an S_{OPT}-scheduling event (OPT scheduling sleep event). An arrival event for M_s is the same as M_1, and an N-scheduling event is the same as a scheduling event for M_1. An S_{OPT}-scheduling event is an event in which an online algorithm A can transmit a packet from a queue, but OPT cannot. Furthermore, A for M_s cannot distinguish between an N-scheduling event and an S_{OPT}-scheduling event. For simplicity, we denote OPT_s an optimal offline algorithm for M_s. Then, we say that OPT_s *sleeps* for A_s if A_s transmits a packet at an S_{OPT}-scheduling event. Note that an online algorithm A_1 for M_1 can be used for M_s. We define p-events and g-events for A_1, OPT_s and an inpit σ_s at M_s in the same way as M_1. For an online algorithm A_1 at M_s, OPT_s and an inpit σ_s at M_s, if an event e is a p-event (g-event, respectively), we write e is a p-event for (A_1, OPT_s, σ_s) (g-event for (A_1, OPT_s, σ_s), respectively).

LEMMA 3.5. *Let A_1 be an online algorithm for M_1 whose competitive ratio is at most c. Let σ_s be any input for M_s in which OPT_s sleeps for A_1 exactly k times, and let σ_1 be an input for M_1 obtained from σ_s by replacing all S_{OPT}-scheduling events by N-scheduling events. Then,* $\min\{(c-1)T_{A_1}(\sigma_1), T_{A_1}(\sigma_1) - k\} \geq \mathcal{G}_{A_1}(\sigma_s)$.

LEMMA 3.6. *Let σ_r be an input for M_r. Then,* $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\} \geq \mathcal{G}_{AS}(\sigma_r)$.

Now, we are ready to show the main lemma in this section.

time	event	ON	OPT	gaps
0		0	0	0
1	a packet arrives	1	1	0
2	a packet arrives	2	2	0
3	a packet arrives	3	3	0
4	a packet arrives	4	4	0
5	Only ON schedules	3	4	0
6	Only OPT schedules	3	3	0
7	Only OPT schedules	3	2	1
8	a packet arrives	4	3	1
9	Only OPT schedules	4	2	2
10	Only OPT schedules	4	1	3
11	a packet arrives	4	2	2
12	Only ON schedules	3	2	1
13	a packet arrives	4	3	1
14	Only OPT schedules	4	2	2
15	a packet arrives	4	3	1
16	a packet arrives	4	4	0
17	Only OPT schedules	4	3	1
18	OPT and ON schedule	3	2	1
19	OPT and ON schedule	2	1	1
20	Only ON schedules	1	1	0
21	OPT and ON schedule	0	0	0

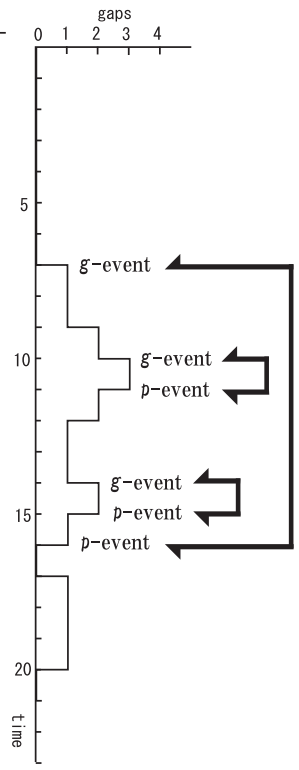

Figure 1: Example of gaps

LEMMA 3.7. *Let σ_r be an input for M_r. Then,* $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\} \geq \mathcal{T}_{B,\alpha}(\sigma_r)$.

PROOF. By Lemma 3.4, $\mathcal{P}_{AS}(\sigma_r) \geq \mathcal{T}_{B,\alpha}(\sigma_r)$. By this inequality and Lemmas 3.3 and 3.6, $\min\{(c-1)R_{AS}(\sigma_r), R_{AS}(\sigma_r) - \mathcal{K}\} \geq \mathcal{T}_{B,\alpha}(\sigma_r)$, which completes the proof. □

3.2.3 Analysis of OS

In this section, we analyze OS to evaluate the number of packets which OPT_r transmits but OS cannot return (Note that the sum of packets returned by OS is different from the sum of packets transmitted by OS). First, we show lemmas about properties of packets which OS and DS store at the same time.

LEMMA 3.8. *Let t be a time when no event occurs. If DS stores a 1-packet p at $Q^{(i)}$ at t, OS also stores p at $Q^{(i)}$ at t.*

LEMMA 3.9. *Let t be a time when no event happens. Then,* $\forall i\ h^{(i)}_{OS}(t) \geq h^{(i)}_{DS}(t)$.

We give some definitions. For an input σ for M_r, $A = \{AS, OS\}$, and a time t when an event does not happen, $R_A(\sigma, t)$ denotes the number of packets returned by A before t, and $T_{OS}(\sigma, t)$ denotes the number of packets transmitted by OS before t.

LEMMA 3.10. *Let t be a time when no event occurs, and σ_r be an input for M_r. Then,* $\forall t\ R_{AS}(\sigma_r, t) + R_{OS}(\sigma_r, t) + \sum_{i=1}^{m} h^{(i)}_{DS}(t) \geq T_{OS}(\sigma_r, t) + \sum_{i=1}^{m} h^{(i)}_{OS}(t)$.

COROLLARY 3.11. *Let σ_r be an input for M_r. Then,* $R_{AS}(\sigma_r) + R_{OS}(\sigma_r) \geq T_{OS}(\sigma_r)$.

We give a definition for the following lemma. Let t be a time when no event happens, and σ_r be an input for M_r. $\mathcal{T}_{\bar{B},1}(\sigma_r, t)$ denotes the number of 1-packets p such that (i) p arrives at $Q^{(i)}$ at time t' where $g^{(i)}_{DS}(t'-) < B$ and DS drops p at $t''(\in [t', t])$, and (ii) OPT_r accepts p at t'. Also, we define a p-event and a g-event for an online algorithm OS, OPT_r and an input σ_r for M_r in the same way as M_1.

LEMMA 3.12. *Let t be a time when no event happens, and σ_r be an input for M_r. Then,* $\mathcal{P}_{OS}(\sigma_r, t) \geq \mathcal{T}_{B,\alpha}(\sigma_r, t) + \mathcal{T}_{\bar{B},1}(\sigma_r, t)$.

In order to evaluate an upper bound on the number of g-events for (OS, OPT_r, σ_r) for an input σ_r at M_r, we consider an extension of M_s (say, $M_{s'}$). For simplicity, we denote $OPT_{s'}$ an optimal offline algorithm for $M_{s'}$. An input for $M_{s'}$ is a sequence of events. An event is an arrival event, an N-scheduling event, an S_{OPT}-scheduling event or an S_{ON}-scheduling event (online algorithm sleep scheduling event). An arrival event, an N-scheduling event, and an S_{OPT}-scheduling event are the same to those for M_s, respectively. An S_{ON}-scheduling event is a counterpart to an S_{OPT}-scheduling event. Namely, an S_{ON}-scheduling event is an event where $OPT_{s'}$ can transmit a packet, but an online algorithm $A_{s'}$ for $M_{s'}$ cannot. Further, $A_{s'}$ cannot know the presence of any S_{ON}-scheduling events. Hence, an online algorithm A_1 for M_1 can be applied for $M_{s'}$ without modification. Then, we say that an online algorithm A_1 for

$M_{s'}$ *sleeps for* $OPT_{s'}$ if A_1 holds a packet at $t-$, a scheduling event happens at t, and $OPT_{s'}$ transmits a packet at an S_{ON}-scheduling event. We define p-events and g-events for an online algorithm A_1, $OPT_{s'}$ and an input $\sigma_{s'}$ at $M_{s'}$ in the same way as M_1. For an online algorithm A_1 at $M_{s'}$, $OPT_{s'}$ and an input σ_s at $M_{s'}$, if an event e is a p-event (g-event, respectively), we write e is a p-event for $(A_1, OPT_{s'}, \sigma_{s'})$ (g-event for $(A_1, OPT_{s'}, \sigma_{s'})$, respectively).

Now, for an online algorithm A_1 at $M_{s'}$, $OPT_{s'}$ and an input σ_s at $M_{s'}$, we are ready to show an upper bound on the number of g-events for $(A_1, OPT_{s'}, \sigma_{s'})$ in the following lemma.

LEMMA 3.13. *Let A_1 be an online algorithm for M_1 whose competitive ratio is at most c. Let $\sigma_{s'}$ be any input for $M_{s'}$ in which $OPT_{s'}$ sleeps for A_1 exactly k times, and A_1 sleeps for $OPT_{s'}$ exactly k' times, and let σ_1 be an input for M_1 obtained from $\sigma_{s'}$ by replacing all S_{OPT}-scheduling events by N-scheduling events (Note that we do not change S_{ON}-scheduling events). Then, $k'+\min\{(c-1)T_{A_1}(\sigma_1), T_{A_1}(\sigma_1)-k\} \geq \mathcal{G}_{A_1}(\sigma_{s'})$.*

Now we are ready to show the lemma which evaluates the number of g-events for (OS, OPT_r, σ_r) for an input σ_r for M_r.

LEMMA 3.14. *Let σ_r be an input for M_r. Then, $R_{AS}(\sigma_r)+\min\{(c-1)(R_{AS}(\sigma_r)+R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geq \mathcal{G}_{OS}(\sigma_r)$.*

LEMMA 3.15. *Let σ_r an input for M_r. Then, $R_{AS}(\sigma_r)+\min\{(c-1)(R_{AS}(\sigma_r)+R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geq \mathcal{T}_{B,\alpha}(\sigma_r)+\mathcal{T}_{\bar{B},1}(\sigma_r)$.*

PROOF. By Lemmas 3.3, 3.12, and 3.14, $R_{AS}(\sigma_r)+\min\{(c-1)(R_{AS}(\sigma_r)+R_{OS}(\sigma_r)), R_{OS}(\sigma_r)\} \geq \mathcal{T}_{B,\alpha}(\sigma_r)+\mathcal{T}_{\bar{B},1}(\sigma_r)$, which completes the proof. □

4. ALGORITHM $SS(A_1)$

In this section, we first give the definition of Simple Scheduling Algorithm(SS) for M_r, which works for the case of small enough α. Then, we evaluate its competitive ratio.

4.1 Simple Scheduling Algorithm(SS)

We give the definition of Simple Scheduling Algorithm (SS). Let A_1 be an online algorithm for M_1. SS uses A_1 as a subroutine, and hence it is written as $SS(A_1)$, but as before, we write "SS" instead of "$SS(A_1)$" when A_1 is clear.

Buffer Management: The definition of the buffer management of SS is exactly the same as that of DS, namely SS accepts packets greedily.
Scheduling: $SS(A_1)$ uses A_1 as a subroutine. SS first considers the input $\sigma(t)$ it has received so far. SS transforms $\sigma(t)$ into $\sigma'''(t)$ by setting a value of all arriving packets to one. It then simulates A_1 on $\sigma'''(t)$, regarding $\sigma'''(t)$ as an input for M_1. Let p be the packet that A_1 decides to transmit at the current scheduling event (namely, at the end of $\sigma'''(t)$). Then, SS returns p.

4.2 Analysis of SS

THEOREM 4.1. *If the competitive ratio of A_1 for M_1 is at most c, then the competitive ratio of $SS(A_1)$ is at most αc.*

PROOF. Let a (b, respectively) be the number of 1-packets (α-packets, respectively) returned by SS. Let a' (b', respectively) be the number of 1-packets (α-packets, respectively) transmitted by OPT_r. Also, let σ_{SS} be an input $\sigma'''(t_F)$ which is converted from σ_r for M_r (Recall that t_F is a time after the last event happens). Then, $R_{SS}(\sigma_{SS}) = a+b$, $V_{SS}(\sigma_r) = a+\alpha b$, $T_{OPT_r}(\sigma_r) = a'+b'$, and $V_{OPT_r}(\sigma_r) = a'+\alpha b'$. Since all packets in an input converted by SS have the same value, $\frac{T_{OPT_r}(\sigma_r)}{R_{SS}(\sigma_{SS})} = \frac{a'+b'}{a+b} \leq c$. Using this inequality, we have that $\frac{V_{OPT_r}(\sigma_r)}{V_{SS}(\sigma_r)} = \frac{a'+\alpha b'}{a+\alpha b} \leq \frac{\alpha(a'+b')}{a+\alpha b} \leq \frac{\alpha c(a+b)}{a+b} = \alpha c$. □

5. COMPETITIVE RATIOS FOR THE MULTI-QUEUE SWITCH MODEL

In this section, we give upper bounds on several variants of M_2, using Theorems 3.1 and 4.1.

COROLLARY 5.1. *There is an online deterministic algorithm for the non-preemptive M_2 whose competitive ratio is at most* 3.177 *for large enough B.*

COROLLARY 5.2. *There is an online deterministic algorithm for the non-preemptive M_2 whose competitive ratio is at most* 3.778 *for $B \geq 2$.*

COROLLARY 5.3. *There is an online deterministic algorithm for the preemptive M_2 whose competitive ratio is at most* 2.465 *for large enough B.*

COROLLARY 5.4. *There is an online deterministic algorithm for the preemptive M_2 whose competitive ratio is at most* 2.577 *for $B \geq 2$.*

COROLLARY 5.5. *There is an online randomized algorithm for the preemptive M_2 whose competitive ratio is at most* 2.214 *for large enough B.*

COROLLARY 5.6. *There is an online randomized algorithm for the preemptive M_2 whose competitive ratio is at most* 2.297 *for any B.*

COROLLARY 5.7. *There is an online randomized algorithm for the non-preemptive M_2 whose competitive ratio is at most* 3.174 *for any B.*

6. 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] S. Albers and M. Schmidt, "On the performance of greedy algorithms in packet buffering," *SIAM J. Comput.*, Vol. 35, No. 2, pp. 278–304, 2005.

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

[4] N. Andelman and Y. Mansour, "Competitive management of non-preemptive queues with multiple values," *In Proc. of the 17th International Symposium on Distributed Computing*, pp. 166–180, 2003.

[5] 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.

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

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

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] A. Kesselman, Y. Mansour and R. Stee, "Improved competitive guarantees for QoS buffering," *In Proc. of the 11th Annual European Symposium on Algorithms*, pp. 361–372, 2003.

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

[19] A. Kesselman, K. Kogan and M. Segal, "Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing," *In Proc. of the 27th ACM symposium on Principles of Distributed Computing*, pp. 335–344, 2008.

[20] A. Kesselman, K. Kogan and M. Segal, "Best effort and priority queuing policies for buffered crossbar switches," *In Proc. of the 15th international colloquium on Structural Information and Communication Complexity*, pp. 170–184, 2008.

[21] A. Kesselman, K. Kogan and M. Segal, "Improved competitive performance bounds for CIOQ switches," *In Proc. of the 16th Annual European Symposium on Algorithms*, pp. 577–588, 2008.

[22] 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.

[23] K. Kobayashi, S. Miyazaki and Y. Okabe, "Competitive buffer management for multi-queue switches in QoS networks using packet buffering algorithms," unpublished manuscript (http://www.net.ist.i.kyoto-u.ac.jp/kobaya/spaa09.pdf.), 2009.

[24] 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,

[25] 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.

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

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