Better Bounds for Online k-Frame Throughput Maximization in Network Switches

Jun Kawahara[1⋆], Koji M. Kobayashi[2⋆⋆], and Shuichi Miyazaki[3⋆⋆⋆]

[1] Graduate School of Information Science, Nara Institute of Science and Technology
jkawahara@is.naist.jp
[2] National Institute of Informatics
kobaya@nii.ac.jp
[3] Academic Center for Computing and Media Studies, Kyoto University
shuichi@media.kyoto-u.ac.jp

Abstract. We consider a variant of the online buffer management problem in network switches, called the k-frame throughput maximization problem (k-FTM). This problem models the situation where a large frame is fragmented into k packets and transmitted through the Internet, and the receiver can reconstruct the frame only if he/she accepts all the k packets. Kesselman et al. introduced this problem and showed that its competitive ratio is unbounded even when $k = 2$. They also introduced an "order-respecting" variant of k-FTM, called k-OFTM, where inputs are restricted in some natural way. They proposed an online algorithm and showed that its competitive ratio is at most $\frac{2kB}{\lfloor B/k \rfloor} + k$ for any $B \geq k$, where B is the size of the buffer. They also gave a lower bound of $\frac{B}{\lfloor 2B/k \rfloor}$ for deterministic online algorithms when $2B \geq k$ and k is a power of 2.

In this paper, we improve upper and lower bounds on the competitive ratio of k-OFTM. Our main result is to improve an upper bound of $O(k^2)$ by Kesselman et al. to $\frac{5B + \lfloor B/k \rfloor - 4}{\lfloor B/2k \rfloor} = O(k)$ for $B \geq 2k$. Note that this upper bound is tight up to a multiplicative constant factor since the lower bound given by Kesselman et al. is $\Omega(k)$. We also give two lower bounds. First we give a lower bound of $\frac{2B}{\lfloor B/(k-1) \rfloor} + 1$ on the competitive ratio of deterministic online algorithms for any $k \geq 2$ and any $B \geq k - 1$, which improves the previous lower bound of $\frac{B}{\lfloor 2B/k \rfloor}$ by a factor of almost four. Next, we present the first nontrivial lower bound on the competitive ratio of randomized algorithms. Specifically, we give a lower bound of $k - 1$ against an oblivious adversary for any $k \geq 3$ and any B. Since a deterministic algorithm, as mentioned above, achieves an upper bound of about $10k$, this indicates that randomization does not help too much.

⋆ The author is supported in part by JSPS KAKENHI Grant Number 23700001.
⋆⋆ The author is supported by Funding Program for World-Leading Innovative R&D on Science and Technology (First Program).
⋆⋆⋆ The author is supported in part by JSPS KAKENHI Grant Number 24500013.

1 Introduction

When transmitting data through the Internet, each data is fragmented into smaller pieces, and such pieces are encapsulated into data packets. Packets are transmitted to the receiver via several switches and routers over a network, and are reconstructed into the original data at the receiver's side. One of the bottlenecks in achieving high throughput is processing ability of switches and routers. If the arrival rate of packets exceeds the processing rate of a switch, some packets must be dropped. To ease this inconvenience, switches are usually equipped with FIFO buffers that temporarily store packets which will be processed later. In this case, the efficiency of buffer management policies is important since it affects the performance of the overall network.

Aiello et al. [1] initiated the analysis of buffer management problem using the *competitive analysis* [10, 32]: An input of the problem is a sequence of events where each event is an arrival event or a send event. At an arrival event, one packet arrives at an input port of the buffer (FIFO queue). Each packet is of unit size and has a positive value that represents its priority. A buffer can store at most B packets simultaneously. At an arrival event, if the buffer is full, the new packet is rejected. If there is room for the new packet, an online algorithm determines whether to accept it or not without knowing the future events. At each send event, the packet at the head of the queue is transmitted. The gain of an algorithm is the sum of the values of the transmitted packets, and the goal of the problem is to maximize it. If, for any input σ, the gain of an online algorithm ALG is at least $1/c$ of the gain of an optimal offline algorithm for σ, then we say that ALG is c-competitive.

Following the work of Aiello et al. [1], there has been a great amount of work related to the competitive analysis of buffer management. For example, Andelman et al. [5] generalized the two-value model of [1] into the multi-value model in which the priority of packets can take arbitrary values. Another generalization is to allow *preemption*, i.e., an online algorithm can discard packets existing in the buffer. Results of the competitiveness on these models are given in [18, 33, 20, 4, 3, 12]. Also, management policies not only for a single queue but also for the whole switch are extensively studied, which includes multi-queue switches [7, 5, 2, 6, 28, 9], shared-memory switches [14, 19, 27], CIOQ switches [21, 8, 25, 22], and crossbar switches [23, 24]. See [13] for a comprehensive survey.

Kesselman et al. [26] proposed another natural extension, called the *k-frame throughput maximization* problem (k-FTM), motivated by a scenario of reconstructing the original data from data packets at the receiver's side. In this model, a unit of data, called a *frame*, is fragmented into k packets (where the jth packet of the frame is called a j-packet for $j \in [1, k]$) and transmitted through the Internet. At the receiver's side, if all the k packets (i.e., the j-packet of the frame for all j) are received, the frame can be reconstructed (in such a case, we say that the frame is *completed*); otherwise, even if one of them is missing, the receiver can obtain nothing. The goal is to maximize the number of completed frames. Kesselman et al. [26] considered this scenario on a single FIFO queue. They first showed that the competitive ratio of any deterministic algorithm for

k-FTM is unbounded even when $k = 2$ (which can also be applied to randomized algorithms with a slight modification). However, their lower bound construction somehow deviates from the real-world situation, that is, although each packet generally arrives in order of departure in a network such as a TCP/IP network, in their adversarial input sequence the 1-packet of the frame f_i arrives prior to that of the frame $f_{i'}$, while the 2-packet of $f_{i'}$ arrives before that of f_i. Motivated by this, they introduced a natural setting for the input sequence, called the *order-respecting* adversary, in which, roughly speaking, the arrival order of the j-packets of f_i and $f_{i'}$ must obey the arrival order of the j'-packets of f_i and $f_{i'}$ ($j' < j$) (a formal definition will be given in Sec. 2). We call this restricted problem the *order-respecting k-frame throughput maximization* problem (k-OFTM). For k-OFTM, they showed that the competitive ratio of any deterministic algorithm is at least $B/\lfloor 2B/k \rfloor$ when $2B \geq k$ and k is a power of 2. As for an upper bound, they designed a non-preemptive algorithm called StaticPartitioning (SP), and showed that its competitive ratio is at most $\frac{2kB}{\lfloor B/k \rfloor} + k$ for any $B \geq k$.

1.1 Our Results

In this paper, we present the following results:

(i) We design a deterministic algorithm Middle-Drop and Flush (MF) for $B \geq 2k$, and show that its competitive ratio is at most $\frac{5B+\lfloor B/k \rfloor -4}{\lfloor B/2k \rfloor}$. Note that this ratio is $O(k)$, which improves $O(k^2)$ of Kesselman et al. [26] and matches the lower bound of $\Omega(k)$ up to a constant factor.

(ii) For any deterministic algorithm, we give a lower bound of $\frac{2B}{\lfloor B/(k-1) \rfloor} + 1$ on the competitive ratio for any $k \geq 2$ and any $B \geq k-1$. This improves the previous lower bound of $\frac{B}{\lfloor 2B/k \rfloor}$ by a factor of almost four. Moreover, we show that the competitive ratio of any deterministic online algorithm is unbounded if $B \leq k-2$.

(iii) In the randomized setting, we establish the first nontrivial lower bound of $k-1$ against an oblivious adversary for any $k \geq 3$ and any B. This bound matches our deterministic upper bound mentioned in (i) up to a constant factor, which implies that randomization does not help for this problem.

Because of the space restriction, all the proofs of the lemmas and theorems are omitted and are included in [17].

1.2 Used Techniques

Let us briefly explain an idea behind our algorithm MF. The algorithm SP by Kesselman et al. [26] works as follows: (1) It virtually divides its buffer evenly into k subbuffers, each with size $A = \lfloor \frac{B}{k} \rfloor$, and each subbuffer (called *j-subbuffer* for $j \in [1, k]$) is used for storing only j-packets. (2) If the j-subbuffer overflows, i.e., if a new j-packet arrives when A j-packets are already stored in the j-subbuffer, it rejects the newly arriving j-packet (the "tail-drop" policy). It can be shown that SP behaves poorly when a lot of j-packets arrive at a burst, which increases SP's competitive ratio as bad as $\Omega(k^2)$ (such a bad example for

SP is included in the full version of this paper [17]). In this paper, we modify the tail-drop policy and employ the “middle-drop” policy, which preempts the $(\lfloor A/2 \rfloor + 1)$st packet in the j-subbuffer and accepts the newly arriving j-packet, which is crucial in improving the competitive ratio to $O(k)$, as explained in the following.

MF partitions the whole set of given frames into *blocks* $BL_1, BL_2, \ldots$, each with about $3B$ frames, using the rule concerning the arrival order of 1-packets. (This rule is explained in Sec. 3.1 at the definition of MF, where the block BL_i corresponds to the set of frames with the *block number* i.) Each block is categorized into *good* or *bad*: At the beginning of the input, all the blocks are good. At some moment during the execution of MF, if there is no more possibility of completing at least $\lfloor A/2 \rfloor$ frames of a block BL_i (as a result of preemptions and/or rejections of packets in BL_i), then BL_i turns bad. In such a case, MF completely gives up BL_i and preempts all the packets belonging to BL_i in its buffer if any (which is called the “flush” operation). Note that at the end of input, MF completes at least $\lfloor A/2 \rfloor$ frames of a good block.

Consider the moment when the block BL_i turns bad from good, which can happen only when preempting a j-packet p (for some j) of BL_i from the j-subbuffer. Due to the property of the middle-drop policy, we can show that there exist two integers i_1 and i_2 ($i_1 < i < i_2$) such that (i) just after this flush operation, BL_{i_1} and BL_{i_2} are good and all the blocks $BL_{i_1+1}, BL_{i_1+2}, \ldots, BL_{i_2-1}$ are bad, and (ii) just before this flush operation, all the j-packets of BL_i (including p) each of which belongs to a frame that still has a chance of being completed are located between p_1 and p_2, where p_1 and p_2 are j-packets in the buffer belonging to BL_{i_1} and BL_{i_2}, respectively. The above (ii) implies that even though i_2 may be much larger than i_1 (and hence there may be many blocks between BL_{i_1} and BL_{i_2}), the arrival times of p_1 and p_2 are close (since p_1 is still in the buffer when p_2 arrived). This means that j-packets of BL_{i_1} through BL_{i_2} arrived at a burst within a very short span, and hence any algorithm (even an optimal offline algorithm OPT) cannot accept many of them. In this way, we can bound the number of packets accepted by OPT (and hence the number of frames completed by OPT) between two consecutive good blocks. More precisely, if BL_{i_1} and BL_{i_2} are consecutive good blocks at the end of the input, we can show that the number of frames in $BL_{i_1}, BL_{i_1+1}, \ldots, BL_{i_2-1}$ completed by OPT is at most $5B + A - 4 = O(B)$ using (i). Recall that MF completes at least $\lfloor A/2 \rfloor = \Omega(B/k)$ frames of BL_{i_1} since BL_{i_1} is good, which leads to the competitive ratio of $O(k)$.

1.3 Related Results

In addition to the above mentioned results, Kesselman et al. [26] proved that for any B, the competitive ratio of a preemptive greedy algorithm for k-OFTM is unbounded when $k \geq 3$. They also considered offline version of k-FTM and proved the approximation hardness. Recently, Kawahara and Kobayashi [16] proved that the optimal competitive ratio of 2-OFTM is 3, which is achieved by a greedy algorithm.

Scalosub et al. [31] proposed a generalization of k-FTM, called the *max frame goodput* problem. In this problem, a set of frames constitute a *stream*, and a constraint is imposed on the arrival order of packets within the same stream. They established an $O((kMB + M)^{k+1})$-competitive deterministic algorithm, where M denotes the number of streams. Furthermore, they showed that the competitive ratio of any deterministic algorithm is $\Omega(kM/B)$.

Emek et al. [11] introduced the *online set packing* problem. This problem is different from k-FTM in that each frame may consist of different number (at most $k_{\max}$) of packets. Also, a frame f consisting of $s(f)$ packets can be reconstructed if $s(f)(1-\beta)$ packets are transmitted, where β ($0 \leq \beta < 1$) is a given parameter. There is another parameter c representing the capacity of a switch. At an arrival event, several packets arrive at an input port of the queue. A switch can transmit c of them instantly, and operates a buffer management algorithm for the rest of the packets, that is, decides whether to accept them (if any). Emek et al. designed a randomized algorithm PRIORITY, and showed that it is $k_{\max}\sqrt{\sigma_{\max}}$-competitive when $\beta = 0$ and $B = 0$, where $\sigma_{\max}$ is the maximum number of packets arriving simultaneously. They also derived a lower bound of $k_{\max}\sqrt{\sigma_{\max}}(\log\log k/\log k)^2$ for any randomized algorithm. If the number of packets in any frame is exactly k, Mansour et al. [29] showed that for any β the competitive ratio of PRIORITY is $8k\sqrt{\sigma_{\max}(1-\beta)/c}$. Moreover, some variants of this problem have been studied [15, 30].

2 Model Description and Notation

In this section, we give a formal description of the *order-respecting k-frame throughput maximization* problem (k-OFTM). A *frame* f consists of k packets $p_1, \ldots, p_k$. We say that two packets p and q belonging to the same frame are *corresponding*, or p *corresponds to* q. There is one buffer (FIFO queue), which can store at most B packets simultaneously. An input is a sequence of *phases* starting from the 0th phase. The ith phase consists of the ith *arrival subphase* followed by the ith *delivery subphase*. At an arrival subphase, some packets arrive at the buffer, and the task of an algorithm is to decide for each arriving packet p, whether to *accept* p or *reject* p. An algorithm can also discard a packet p' existing in the current buffer in order to make space (in which case we say that the algorithm *preempts* p'). If a packet p is rejected or preempted, we say that p is *dropped*. If a packet is accepted, it is stored at the tail of the queue. Packets accepted at the same arrival subphase can be inserted into the queue in an arbitrary order. At a delivery subphase, the first packet of the queue is transmitted if the buffer is nonempty. For a technical reason, we consider only the inputs in which at least one packet arrives.

If a packet p arrives at the ith arrival subphase, we write $\mathrm{arr}(p) = i$. For any frame $f = \{p_1, \ldots, p_k\}$ such that $\mathrm{arr}(p_1) \leq \cdots \leq \mathrm{arr}(p_k)$, we call p_i the *i-packet* of f. Consider two frames $f_i = \{p_{i,1}, \ldots, p_{i,k}\}$ and $f_{i'} = \{p_{i',1}, \ldots, p_{i',k}\}$ such that $\mathrm{arr}(p_{i,1}) \leq \cdots \leq \mathrm{arr}(p_{i,k})$ and $\mathrm{arr}(p_{i',1}) \leq \cdots \leq \mathrm{arr}(p_{i',k})$. If for any j and j', $\mathrm{arr}(p_{i,j}) \leq \mathrm{arr}(p_{i',j})$ if and only if $\mathrm{arr}(p_{i,j'}) \leq \mathrm{arr}(p_{i',j'})$, then we say that f_i

and $f_{i'}$ are *order-respecting*. If any two frames in an input sequence σ are order-respecting, we say that σ is *order-respecting*. If all the packets constituting a frame f are transmitted, we say that f is *completed*, otherwise, f is *incompleted*. The goal of k-FTM is to maximize the number of completed frames. k-OFTM is k-FTM where inputs are restricted to order-respecting sequences.

For an input σ, the *gain* of an algorithm ALG is the number of frames completed by ALG and is denoted by $V_{ALG}(\sigma)$. If ALG is a randomized algorithm, the gain of ALG is defined as an expectation $\mathbb{E}[V_{ALG}(\sigma)]$, where the expectation is taken over the randomness inside ALG. If $V_{ALG}(\sigma) \geq V_{OPT}(\sigma)/c$ ($\mathbb{E}[V_{ALG}(\sigma)] \geq V_{OPT}(\sigma)/c$) for an arbitrary input σ, we say that ALG is *c-competitive*, where OPT is an optimal offline algorithm for σ. Without loss of generality, we can assume that OPT never preempts packets and never accepts a packet of an incompleted frame.

3 Upper Bound

In this section, we present our algorithm Middle-Drop and Flush (MF) and analyze its competitive ratio.

3.1 Algorithm

We first give notation needed to describe MF. Suppose that n packets $p_1, p_2, \dots, p_n$ arrive at MF's buffer at the ith arrival subphase. For each packet, MF decides whether to accept it or not one by one (in some order defined later). Let t_{p_j} denote the time when MF deals with the packet p_j, and let us call t_{p_j} the *decision time* of p_j. Hence if $p_1, p_2, \dots, p_n$ are processed in this order, we have that $t_{p_1} < t_{p_2} < \cdots < t_{p_n}$. (We assume that OPT also deals with p_j at the same time t_{p_j}, which makes the competitive analysis simpler.) Also, let us call the time when MF transmits a packet from the head of its buffer at the ith delivery subphase the *delivery time* of the ith delivery subphase. A decision time or a delivery time is called an *event time*, and any other moment is called a *non-event time*. Note that during the non-event time, the configuration of the buffer is unchanged. For any event time t, $t+$ denotes any non-event time between t and the next event time. Similarly, $t-$ denotes any non-event time between t and the previous event time.

Let ALG be either MF or OPT. For a non-event time t and a packet p of a frame f, we say that p is *valid* for ALG at t if ALG has not dropped any packet of f before t, i.e., f still has a chance of being completed. In this case we also say that the frame f is *valid* for ALG at t. Note that a completed frame is valid at the end of the input. For a j-packet p and a non-event time t, if p is stored in MF's buffer at t, we define $\ell(t, p)$ as "1+(the number of j-packets located in front of p)", that is, p is the $\ell(t, p)$th j-packet in MF's queue. If p has not yet arrived at t, we define $\ell(t, p) = \infty$.

During the execution, MF virtually runs the following greedy algorithm GR_1 on the same input sequence. Roughly speaking, GR_1 is greedy for only 1-packets and ignores all $j(\geq 2)$-packets. Formally, GR_1 uses a FIFO queue of the same size B. At an arrival of a packet p, GR_1 rejects it if it is a j-packet for $j \geq 2$. If p is a 1-packet, GR_1 accepts it whenever there is a space in the queue. At a delivery subphase, GR_1 transmits the first packet of the queue as usual.

MF uses two internal variables `Counter` and `Block`. `Counter` is used to count the number of packets accepted by GR_1 modulo $3B$. `Block` takes a positive integer value; it is initially one and is increased by one each time `Counter` is reset to zero.

Define $A = \lfloor B/k \rfloor$. MF stores at most A j-packets for any j. For $j = 1$, MF refers to the behavior of GR_1 in the following way: Using two variables `Counter` and `Block`, MF divides 1-packets accepted by GR_1 into blocks according to their arrival order, each with $3B$ 1-packets. MF accepts the first A packets of each block and rejects the rest. For $j \geq 2$, MF ignores j-packets that are not valid. When processing a valid j-packet p, if MF already has A j-packets in its queue, then MF preempts the one in the "middle" among those j-packets and accepts p.

For a non-event time t, let $b(t)$ denote the value of `Block` at t. For a packet p, we define the *block number* $g(p)$ of p as follows. For a 1-packet p, $g(p) = b(t-)$ where t is the decision time of p, and for some $j(\geq 2)$ and a j-packet p , $g(p) = g(p')$ where p' is the 1-packet corresponding to p. Hence, all the packets of the same frame have the same block number. We also define the block number of frames in a natural way, namely, the block number $g(f)$ of a frame f is the (unique) block number of the packets constituting f. For a non-event time t and a positive integer u, let $h_{ALG,u}(t)$ denote the number of frames f valid for ALG at t such that $g(f) = u$.

Recall that at an arrival subphase, more than one packet may arrive at a queue. MF processes the packets ordered non-increasingly first by their frame indices and then by block numbers. If both are equal, they are processed in arbitrary order. That is, MF processes these packets by the following rule: Consider an i-packet p and an i'-packet p'. If $i < i'$, p is processed before p' and if $i' < i$, p' is processed before p. If $i = i'$, then p is processed before p' if $g(p) < g(p')$ and p' is processed before p if $g(p') < g(p)$. If $i = i'$ and $g(p) = g(p')$, the processing order is arbitrary. The formal description of MF is as follows.

Middle-Drop and Flush

Initialize: `Counter` $:= 0$, `Block` $:= 1$.
Let p be a j-packet to be processed.
Case 1: $j = 1$:
- **Case 1.1:** If GR_1 rejects p, reject p.
- **Case 1.2:** If GR_1 accepts p, set `Counter` $:=$ `Counter` $+1$ and do the following.
  - **Case 1.2.1:** If `Counter` $\leq A$, accept p. (We can guarantee that MF's buffer has a space whenever `Counter` $\leq A$, as proven in [17].)
  - **Case 1.2.2:** If $A <$ `Counter` $< 3B$, reject p.
  - **Case 1.2.3:** If `Counter` $= 3B$, reject p and set `Counter` $:= 0$ and `Block` $:=$ `Block` $+ 1$.

Case 2: $j \geq 2$:
Case 2.1: If p is not valid for MF at t_p-, reject p.
Case 2.2: If p is valid for MF at t_p-, do the following.
Case 2.2.1: If the number of j-packets in MF's buffer at t_p- is at most $A-1$, accept p.
Case 2.2.2: If the number of j-packets in MF's buffer at t_p- is at least A, then preempt the j-packet p' such that $\ell(t_p-, p') = \lfloor A/2 \rfloor + 1$, and accept p. Preempt all the packets corresponding to p' (if any).
Case 2.2.2.1: If $h_{MF,g(p')}(t_p-) \leq \lfloor A/2 \rfloor$, preempt all the packets p'' in MF's buffer such that $g(p'') = g(p')$. (Call this operation "flush".)
Case 2.2.2.2: If $h_{MF,g(p')}(t_p-) \geq \lfloor A/2 \rfloor + 1$, do nothing.

3.2 Overview of the Analysis

Let τ be any fixed time after MF processes the final event, and let c denote the value of `Counter` at τ. Also, we define $M = b(\tau) - 1$ if $c = 0$ and $M = b(\tau)$ otherwise. Note that for any frame f, $1 \leq g(f) \leq M$. Define the set G of integers as $G = \{M\} \cup \{i \mid$ there are at least $\lfloor A/2 \rfloor$ frames f completed by MF such that $g(f) = i\}$ and let $m = |G|$. For each $j \in [1, m]$, let a_j be the jth smallest integer in G. We call a block number *good* if it is in G and *bad* otherwise. Note that a_j denotes the jth good block number, and in particular that $a_m = M$ since $M \in G$. Our first key lemma is the following:

Lemma 1. $a_1 = 1$.

Since at the end of the input any valid frame is completed, we have $V_{OPT}(\sigma) = \sum_{i=1}^{M} h_{OPT,i}(\tau)$ and $V_{MF}(\sigma) = \sum_{i=1}^{M} h_{MF,i}(\tau) \geq \sum_{i=1}^{m} h_{MF,a_i}(\tau)$.

We first bound the gain of MF for good block numbers, which follows from the definition of G:

$$h_{MF,a_i}(\tau) \geq \lfloor A/2 \rfloor \text{ for any } i \in [1, m-1]. \quad (1)$$

We next focus on the mth good block number M. Since it has some exceptional properties, we discuss the number of completed frames with block number M independently of the other good block numbers as follows:

Lemma 2. *(a) If either $c = 0$ or $c \in [\lfloor A/2 \rfloor, 3B-1]$, $h_{MF,M}(\tau) \geq \lfloor A/2 \rfloor$. (b) If $c \in [1, \lfloor A/2 \rfloor - 1]$ and $M \geq 2$, $h_{MF,M}(\tau) + B - 1 \geq h_{OPT,M}(\tau)$. (c) If $c \in [1, \lfloor A/2 \rfloor - 1]$ and $M = 1$, $h_{MF,M}(\tau) \geq h_{OPT,M}(\tau)$.*

Also, we evaluate the number of OPT's completed frames from a viewpoint of good block numbers:

Lemma 3. *(a) $h_{OPT,M}(\tau) \leq 4B - 1$. (b) $\sum_{j=a_1}^{a_2-1} h_{OPT,j}(\tau) \leq 4B + A - 3$. (c) $\sum_{j=a_i}^{a_{i+1}-1} h_{OPT,j}(\tau) \leq 5B + A - 4$ for any $i \in [2, m-1]$.*

Using the above inequalities, we can obtain the competitive ratio of MF by case analysis on the values of M and c. First, note that if $M = 1$ then $c \geq 1$ because at least one packet arrives. Thus $V_{OPT}(\sigma) > 0$. Now if $M = 1$ and $c \in [1, \lfloor A/2 \rfloor - 1]$, then $\frac{V_{OPT}(\sigma)}{V_{MF}(\sigma)} = \frac{h_{OPT,1}(\tau)}{h_{MF,1}(\tau)} \leq 1$ by Lemma 2 (c). If $M = 1$ and $c \in [\lfloor A/2 \rfloor, 3B - 1]$, then $\frac{V_{OPT}(\sigma)}{V_{MF}(\sigma)} = \frac{h_{OPT,1}(\tau)}{h_{MF,1}(\tau)} \leq \frac{4B-1}{\lfloor A/2 \rfloor} < \frac{5B+A-4}{\lfloor A/2 \rfloor}$ by Lemma 2(a) and Lemma 3(a).

If $M \geq 2$ and $c \in \{0\} \cup [\lfloor A/2 \rfloor, 3B - 1]$,

$$\begin{aligned} V_{OPT}(\sigma) &= \sum_{i=1}^{M} h_{OPT,i}(\tau) = \sum_{i=1}^{m-1} \sum_{j=a_i}^{a_{i+1}-1} h_{OPT,j}(\tau) + h_{OPT,a_m}(\tau) \\ &\leq (m-1)(5B + A - 4) - B + 1 + (4B - 1) < m(5B + A - 4) \end{aligned}$$

by Lemma 3 (note that $a_1 = 1$ by Lemma 1 and $a_m = M$). Also, $V_{MF}(\sigma) \geq \sum_{i=1}^{m} h_{MF,a_i}(\tau) \geq m\lfloor A/2 \rfloor$ by (1) and Lemma 2(a). Therefore, $\frac{V_{OPT}(\sigma)}{V_{MF}(\sigma)} < \frac{5B+A-4}{\lfloor A/2 \rfloor}$. Finally, if $M \geq 2$ and $c \in [1, \lfloor A/2 \rfloor - 1]$,

$$\begin{aligned} V_{OPT}(\sigma) &= \sum_{i=1}^{M} h_{OPT,i}(\tau) = \sum_{i=1}^{m-1} \sum_{j=a_i}^{a_{i+1}-1} h_{OPT,j}(\tau) + h_{OPT,a_m}(\tau) \\ &\leq (m-1)(5B + A - 4) - B + 1 + h_{OPT,M}(\tau) \\ &\leq (m-1)(5B + A - 4) + h_{MF,M}(\tau) \end{aligned}$$

by Lemma 2(b) and Lemma 3(b) and (c). Also, $V_{MF}(\sigma) = \sum_{i=1}^{m} h_{MF,a_i}(\tau) \geq (m-1)\lfloor A/2 \rfloor + h_{MF,M}(\tau)$ by (1). Therefore,

$$\frac{V_{OPT}(\sigma)}{V_{MF}(\sigma)} \leq \frac{(m-1)(5B + A - 4) + h_{MF,M}(\tau)}{(m-1)\lfloor A/2 \rfloor + h_{MF,M}(\tau)} < \frac{5B + A - 4}{\lfloor A/2 \rfloor}.$$

We have proved that in all the cases $\frac{V_{OPT}(\sigma)}{V_{MF}(\sigma)} < \frac{5B+A-4}{\lfloor A/2 \rfloor}$. By noting that $\frac{5B+A-4}{\lfloor A/2 \rfloor} = \frac{5B+\lfloor B/k \rfloor-4}{\lfloor B/2k \rfloor}$, we have the following theorem:

Theorem 1. *When $B/k \geq 2$, the competitive ratio of MF is at most $\frac{5B+\lfloor B/k \rfloor-4}{\lfloor B/2k \rfloor}$.*

4 Lower Bound for Deterministic Algorithms

In this section, we give a lower bound on the competitive ratio for deterministic algorithms, improving the previous lower bound by a constant factor.

Theorem 2. *Suppose that $k \geq 2$. The competitive ratio of any deterministic algorithm is at least $\frac{2B}{\lfloor B/(k-1) \rfloor} + 1$ if $B \geq k - 1$, and unbounded if $B \leq k - 2$.*

5 Lower Bound for Randomized Algorithms

As for randomized algorithms, we give a first nontrivial lower bound. As mentioned previously, this matches the upper bound we proved in Sec. 3.2 up to a constant factor, implying that randomization does not help too much.

Theorem 3. *When $k \geq 3$, the competitive ratio of any randomized algorithm is at least $k - 1 - \epsilon$ for any constant ϵ against an oblivious adversary.*

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 Journal on Computing*, Vol. 35, No. 2, pp. 278–304, 2005.
3. N. Andelman, "Randomized queue management for DiffServ," *In Proc. of the 17th 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 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, "An improved algorithm for CIOQ switches," *ACM Transactions on Algorithms*, Vol. 2, No. 2, pp. 282–295, 2006,
9. M. Bienkowski, "An optimal lower bound for buffer management in multi-queue switches," *In Proc. of the 16th ACM-SIAM Symposium on Discrete Algorithms*, pp. 1295–1305, 2010.
10. A. Borodin and R. El-Yaniv, "Online computation and competitive analysis," *Cambridge University Press*, 1998.
11. Y. Emek, M. Halldórsson, Y. Mansour, B. Patt-Shamir, J. Radhakrishnan and D. Rawitz, "Online set packing and competitive scheduling of multi-part tasks," *In Proc. of the 29th ACM symposium on Principles of Distributed Computing*, pp. 440–449, 2010.
12. M. Englert and M. Westermann, "Lower and upper bounds on FIFO buffer management in QoS switches," *In Proc. of the 14th European Symposium on Algorithms*, pp. 352–363, 2006.
13. M. Goldwasser, "A survey of buffer management policies for packet switches," ACM SIGACT News, Vol.41, No. 1, pp. 100–128, 2010.
14. E. Hahne, A. Kesselman and Y. Mansour, "Competitive buffer management for shared-memory switches," *In Proc. of the 13th ACM Symposium on Parallel Algorithms and Architectures*, pp. 53–58, 2001.
15. M. Halldórsson, B. Patt-Shamir and D. Rawitz, "Online Scheduling with Interval Conflicts," *In Proc. of the 28th Symposium on Theoretical Aspects of Computer Science*, pp. 472–483, 2011.
16. J. Kawahara, and K. M. Kobayashi, "Optimal Buffer Management for 2-Frame Throughput Maximization," *In Proc. of the 20th international colloquium on Structural Information and Communication Complexity*, 2013.
17. J. Kawahara, K. M. Kobayashi and S. Miyazaki, "Better Bounds for Online k-Frame Throughput Maximization in Network Switches," *arXiv*:1309.4919 [cs.DS], 2013.
18. A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko, "Buffer overflow management in QoS switches," *SIAM Journal on Computing*, Vol. 33, No. 3, pp. 563–583, 2004.

19. 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.
20. A. Kesselman, Y. Mansour and R. Stee, "Improved competitive guarantees for QoS buffering," *In Proc. of the 11th European Symposium on Algorithms*, pp. 361–372, 2003.
21. A. Kesselman and A. Rosén, "Scheduling policies for CIOQ switches," *Journal of Algorithms*, Vol. 60, No. 1, pp. 60–83, 2006.
22. A. Kesselman and A. Rosén, "Controlling CIOQ switches with priority queuing and in multistage interconnection networks," *Journal of Interconnection Networks*, Vol. 9, No. 1/2, pp. 53–72, 2008.
23. 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.
24. 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.
25. A. Kesselman, K. Kogan and M. Segal, "Improved competitive performance bounds for CIOQ switches," *In Proc. of the 16th European Symposium on Algorithms*, pp. 577–588, 2008.
26. A. Kesselman, B. Patt-Shamir and G. Scalosub, "Competitive buffer management with packet dependencies," *In Proc. of the 23rd IEEE International Parallel and Distributed Processing Symposium*, pp. 1–12, 2009.
27. 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 ACM Symposium on Parallel Algorithms and Architectures*, pp. 358–364, 2007.
28. K. Kobayashi, S. Miyazaki and Y. Okabe, "Competitive buffer management for multi-queue switches in QoS networks using packet buffering algorithms," *In Proc. of the 21st ACM Symposium on Parallel Algorithms and Architectures*, pp. 328–336, 2009.
29. Y. Mansour, B. Patt-Shamir, and D. Rawitz, "Overflow management with multipart packets," *In Proc. of the 31st IEEE Conference on Computer Communications*, pp. 2606–2614, 2011.
30. Y. Mansour, B. Patt-Shamir, and D. Rawitz, "Competitive router scheduling with structured data," *In Proc. of the 9th Workshop on Approximation and Online Algorithms*, pp. 219–232, 2011.
31. G. Scalosub, P. Marbach and J. Liebeherr, "Buffer management for aggregated streaming data with packet dependencies," *In Proc. of the 29th IEEE Conference on Computer Communications*, pp. 1–5, 2010.
32. 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.
33. M. Sviridenko, "A lower bound for on-line algorithms in the FIFO model," unpublished manuscript, 2001.