

# Better Bounds for Online $k$ -Frame Throughput Maximization in Network Switches \*

Jun Kawahara<sup>1</sup>, Koji M. Kobayashi<sup>2</sup>, Shuichi Miyazaki<sup>3</sup>

<sup>1</sup>Graduate School of Information Science, Nara Institute of Science and Technology

<sup>2</sup> Corresponding author, National Institute of Informatics,

2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 1018430, Japan, kobaya@nii.ac.jp

<sup>3</sup>Academic Center for Computing and Media Studies, Kyoto University

## 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.

Keywords: Buffer management; Online problem; Competitive Analysis; Packet Fragmentation

## 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

---

\*A preliminary version of this paper was presented at the 24th International Symposium on Algorithms and Computation (ISAAC 2013).

routers. If the arrival rate of packets exceeds the processing rate of a switch, some packets must be dropped. To ease this degradation, 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, 31]: 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  $\sigma$ , the gain of an online algorithm  $ALG$  is at least  $1/c$  of the gain of an optimal offline algorithm for  $\sigma$ , 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 [17, 32, 19, 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, 27, 9], shared-memory switches [14, 18, 26], CIOQ switches [20, 8, 24, 21], and crossbar switches [22, 23]. See [13] for a comprehensive survey.

Kesselman et al. [25] 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  $j$ th 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 every  $j$ ) 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. [25] 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 an 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 more 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. (Note that this ratio is  $\Omega(k)$ .) As for an upper bound, they designed a non-preemptive algorithm called STATICPARTITIONING ( $SP$ ), and showed that it is  $\frac{2kB}{\lfloor B/k \rfloor} + k = O(k^2)$ -competitive for any  $B \geq k$ . They conjecture that the true competitive

ratio is  $\Theta(k)$ .

## 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  $\frac{5B + \lfloor B/k \rfloor - 4}{\lfloor B/2k \rfloor}$ . This ratio is  $O(k)$ , which improves  $O(k^2)$  of Kesselman et al. [25] and matches the lower bound of  $\Omega(k)$  up to a constant factor. Hence we have solved their conjecture affirmatively.

(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 too much for this problem.

## 1.2 Used Techniques

Let us briefly explain an idea behind our algorithm *MF*. The algorithm *SP* by Kesselman et al. [25] 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 given in Appendix A).

In this paper, we introduce two major ideas to develop a better algorithm. One is to 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. The other is to preempt all of the packets satisfying some conditions at some moment, which we call the “flush” operation. The two ideas are crucial in improving the competitive ratio to  $O(k)$ , as explained in the following. *MF* partitions the whole set of given frames into *blocks*  $BL_1, BL_2, \dots$ , 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  $BL_{i_1+1}, BL_{i_1+2}, \dots, BL_{i_2-1}$  are bad, and (ii) just before this flush operation, there exist two *j*-packets  $p_1$  and  $p_2$  in the

buffer such that  $p_1$  and  $p_2$  belong to  $BL_{i_1}$  and  $BL_{i_2}$ , respectively, and (iii) 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  $p_2$ . 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  $BL_{i_1}, BL_{i_1+1}, \dots, BL_{i_2-1}$  completed by  $OPT$  is at most  $5B + A - 4 = O(B)$  using (i), (ii) and (iii). 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  $O(k)$ . (The formal definitions of good and bad blocks are given in Sec. 3.2 and the proof of Lemma 3.9.)

### 1.3 Related Results

In addition to the above mentioned results, Kesselman et al. [25] 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 the offline version of  $k$ -FTM and showed some approximation hardness results. 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. [30] proposed a generalization of  $k$ -FTM, called the *max frame goodput* problem. In this problem, a set of frames constitutes 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 and pointed out that the problem is related to  $k$ -FTM. 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  $\beta$  ( $0 \leq \beta < 1$ ) 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. The switch can transmit  $c$  of them instantly, and operates a buffer management algorithm for the rest of the packets (if any), that is, decides whether to accept them. 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. [28] showed that for any  $\beta$  the competitive ratio of **PRIORITY** is  $8k\sqrt{\sigma_{\max}(1 - \beta)}/c$ . Moreover, some variants of this problem have been studied [15, 29].

## 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  $p_1, \dots, p_k$ . 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  $i$ th phase consists of the  $i$ th *arrival subphase* followed by the  $i$ th *delivery subphase*. At an arrival subphase, one or more 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  $i$ th arrival subphase, we write  $\text{arr}(p) = i$ . For every frame  $f = \{p_1, \dots, p_k\}$  such that  $\text{arr}(p_1) \leq \dots \leq \text{arr}(p_k)$ , we call  $p_i$  the  $i$ -packet of  $f$ . Consider two frames  $f_i = \{p_{i,1}, \dots, p_{i,k}\}$  and  $f_{i'} = \{p_{i',1}, \dots, p_{i',k}\}$  such that  $\text{arr}(p_{i,1}) \leq \dots \leq \text{arr}(p_{i,k})$  and  $\text{arr}(p_{i',1}) \leq \dots \leq \text{arr}(p_{i',k})$ . If for every  $j$  and  $j'$ ,  $\text{arr}(p_{i,j}) \leq \text{arr}(p_{i',j})$  if and only if  $\text{arr}(p_{i,j'}) \leq \text{arr}(p_{i',j'})$ , then we say that  $f_i$  and  $f_{i'}$  are *order-respecting*. If every two frames in an input sequence  $\sigma$  are order-respecting, we say that  $\sigma$  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  $\sigma$ , 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 every input  $\sigma$ , we say that  $ALG$  is  $c$ -*competitive*, where  $OPT$  is an optimal offline algorithm for  $\sigma$ . 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  $i$ th 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  $t_{p_1} < t_{p_2} < \dots < t_{p_n}$ . (For convenience, in the later analysis, we assume that  $OPT$  also deals with  $p_j$  at the same time  $t_{p_j}$ .) Also, let us call the time when  $MF$  transmits a packet from the head of its buffer at the  $i$ th delivery subphase the *delivery time* of the  $i$ th 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 before  $p$ )”, 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 the 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 a  $j(\in [2, k])$ -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  $g(f)$  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  $f$  such that  $f$  is valid for  $ALG$  at  $t$  and  $g(f) = u$ .

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. To illustrate an execution of  $MF$ , we give an example in Appendix B.

---

### 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 prove in Lemma 3.5(c) that  $MF$ 's buffer has a space whenever **Counter**  $\leq A$ .)

**Case 1.2.2:** If  $A < \text{Counter} < 3B$ , 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  $\tau$  be any fixed time after  $MF$  processes the final event, and let  $c$  denote the value of **Counter** at  $\tau$ . 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  $G = \{M\} \cup \{i \mid MF \text{ completes at least } \lfloor A/2 \rfloor \text{ frames } f \text{ such that } g(f) = i\}$  and let  $m = |G|$ . We call a block number *good* if it is in  $G$  and *bad* otherwise. For each  $j \in [1, m]$ , let  $a_j$  be the  $j$ th smallest integer in  $G$ . Note that  $a_j$  denotes the  $j$ th good block number, and in particular that  $a_m = M$  since  $M \in G$ . Our first key lemma is the following, saying that the first block is always good:

**Lemma 3.1**  $a_1 = 1$ .

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 guarantee 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  $m$ th 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 3.2** (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 3.3** (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 since at least one packet arrives,  $V_{OPT}(\sigma) > 0$  holds and  $M = 1$  implies  $c \geq 1$ . Now if  $M = 1$  and  $c \in [1, \lfloor A/2 \rfloor - 1]$ ,  $h_{MF,1}(\tau) \geq h_{OPT,1}(\tau)$  by Lemma 3.2(c). Since  $h_{MF,1}(\tau) \geq h_{OPT,1}(\tau) = V_{OPT}(\sigma) > 0$ ,  $\frac{V_{OPT}(\sigma)}{V_{MF}(\sigma)} = \frac{h_{OPT,1}(\tau)}{h_{MF,1}(\tau)} \leq 1$ . 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 3.2(a) and Lemma 3.3(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 3.3 (note that  $a_1 = 1$  by Lemma 3.1 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 Eq. (1) and Lemma 3.2(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 Lemmas 3.2(b), 3.3(b) and 3.3(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 Eq. (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  $\frac{V_{OPT}(\sigma)}{V_{MF}(\sigma)} < \frac{5B+A-4}{\lfloor A/2 \rfloor}$  in all the cases. 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 3.4** *When  $B/k \geq 2$ , the competitive ratio of  $MF$  is at most  $\frac{5B+\lfloor B/k \rfloor - 4}{\lfloor B/2k \rfloor}$ .*

### 3.3 Analysis of $MF$

In this section, we first show the feasibility of Case 1.2.1 of  $MF$ . We then give the proofs of Lemmas 3.1, 3.2, and 3.3 in the subsequent sections.

#### 3.3.1 Feasibility of $MF$

In this section, we guarantee the feasibility of  $MF$  by proving Lemma 3.5(c). At the same time, we prove Lemma 3.5(a) and (b) for later use. If an algorithm  $ALG$  transmits a packet  $p$  at the  $i$ th delivery subphase, we write  $\text{del}_{ALG}(p) = i$ .

**Lemma 3.5** *Suppose that  $GR_1$  accepts  $z(\geq 2B)$  1-packets. Let  $p_i$  ( $i \in [1, z]$ ) denote the  $i$ th 1-packet accepted by  $GR_1$ . Then, the following holds. (a) If  $z \geq 2B$ , then for any  $j(\in [1, z - 2B + 1])$  such that  $MF$  accepts  $p_j$ ,  $\text{del}_{MF}(p_j) < \text{arr}(p_{j+2B-1})$ . (b) For any  $u(\geq 0)$ ,  $MF$  accepts all the 1-packets  $p_{3Bu+1}, \dots, p_{3Bu+A}$ , and their block number is  $u + 1$ . (c) If  $0 \leq \text{Counter} \leq A - 1$  just before the decision time of a 1-packet  $p$ ,  $MF$ 's buffer has a space to accept  $p$ .*

*Proof.* (a) Consider a 1-packet  $p_j$  satisfying the condition of the lemma, and consider the non-event time  $t_{p_j} +$ , i.e., the moment just after  $GR_1$  and  $MF$  accept  $p_j$ . By definition,  $MF$  certainly transmits any 1-packet inserted into its buffer. In addition,  $MF$  will transmit  $p_j$  within  $B$  phases,

since the buffer size is  $B$ , and only one packet can be transmitted in one phase. That is, the number of delivery subphases between  $t_{p_j} +$  and the moment before  $MF$  transmits  $p_j$  is at most  $B - 1$ , which means that  $GR_1$  can also transmit at most  $B - 1$  packets during this period. On the other hand,  $GR_1$  accepts  $p_j$  as well, and there exists at least one packet in  $GR_1$ 's buffer at  $t_{p_j} +$ . Hence there are at most  $B - 1$  vacancies in  $GR_1$ 's buffer at this moment. Therefore, the number of packets  $GR_1$  can accept between  $t_{p_j} +$  and the moment before  $MF$  transmits  $p_j$  is at most  $(B - 1) + (B - 1) = 2B - 2$ . In other words,  $MF$  transmits  $p_j$  before  $GR_1$  accepts  $p_{j+2B-1}$ . This proves  $\text{del}_{MF}(p_j) < \text{arr}(p_{j+2B-1})$ .

(b) First, recall that  $MF$  can always store up to  $A$   $x$ -packets for  $x \in [1, k]$ . Due to Cases 1.2.1, 1.2.2 and 1.2.3,  $MF$  accepts packets  $p_1, p_2, \dots, p_A$ , and rejects  $p_{A+1}, p_{A+2}, \dots, p_{3B}$ . During this period, **Block** stays 1. Just after  $MF$  rejects  $p_{3B}$ , **Block** is incremented to 2, and **Counter** is reset to 0. Since  $\text{del}_{MF}(p_A) < \text{arr}(p_{A+2B-1})$  by the proof of the part (a) of this lemma,  $\text{del}_{MF}(p_A) < \text{arr}(p_{3B+1})$ , which means that there exists no 1-packet in  $MF$ 's queue at the non-event time  $t_{p_{3B+1}} -$ . Hence,  $MF$  starts accepting  $p_{3B+1}, p_{3B+2}, \dots, p_{3B+A}$ . By continuing this argument, we can prove part (b).

(c)  $0 \leq \text{Counter} \leq A - 1$  holds just before the decision times of  $p_{3Bu+1}, \dots, p_{3Bu+A}$  for each  $u = 0, 1, \dots$ . Thus, (c) is immediate from the proof of part (b).  $\square$

### 3.3.2 Proof of Lemma 3.1

To prove Lemma 3.1, we use Lemma 3.5 and the following lemmas. Roughly speaking, Lemma 3.6 says that, for each  $j$ , block numbers of  $j$ -packets are assigned in order of arrival, and Lemma 3.7 says that, for each  $j'$ , packets in the  $j'$ -subbuffer are stored in order of block number.

**Lemma 3.6** (a) Let  $p$  be any 1-packet accepted by  $MF$  and  $q$  be any 1-packet such that  $g(p) < g(q)$ . Then,  $\text{arr}(p) < \text{arr}(q)$ . (b) For  $x \in [2, k]$ , let  $p'$  and  $q'$  be any  $x$ -packets such that  $g(p') < g(q')$  and suppose that  $MF$  accepts the 1-packet corresponding to  $p'$ . Then,  $\text{arr}(p') \leq \text{arr}(q')$ .

*Proof.* (a) Note that  $p$  is accepted by also  $GR_1$ . Let  $\hat{q}$  be the first 1-packet with block number  $g(q)$ . Clearly  $GR_1$  accepts  $\hat{q}$  and  $\text{arr}(\hat{q}) \leq \text{arr}(q)$ . Suppose that  $p$  and  $\hat{q}$  are the  $i$ th and the  $j$ th packets, respectively, accepted by  $GR_1$ . By Lemma 3.5(b) and the assumption that  $g(p) < g(q) (= g(\hat{q}))$ ,  $j - i \geq (3B(g(\hat{q}) - 1) + 1) - (3B(g(p) - 1) + A) = 3B(g(\hat{q}) - g(p)) + 1 - A \geq 3B + 1 - A > 2B$ . Then by Lemma 3.5(a),  $p$  is transmitted by  $MF$  before  $\hat{q}$  arrives. Therefore,  $\text{arr}(p) < \text{arr}(\hat{q})$ , which means that  $\text{arr}(p) < \text{arr}(q)$ . This completes the proof.

(b) Let  $p_1$  and  $q_1$  be the 1-packets corresponding to  $p'$  and  $q'$ , respectively. Since  $g(p') < g(q')$ ,  $g(p_1) < g(q_1)$ . Therefore,  $\text{arr}(p_1) < \text{arr}(q_1)$  by (a). Since the input is order-respecting,  $\text{arr}(p') \leq \text{arr}(q')$ .  $\square$

**Lemma 3.7** Let  $t$  be a non-event time. For any  $x \in [1, k]$ , let  $p$  be an  $x$ -packet stored in  $MF$ 's buffer at  $t$ , and let  $q$  be an  $x$ -packet which is stored in  $MF$ 's buffer at  $t$ . If  $\ell(t, p) < \ell(t, q)$ , then  $g(p) \leq g(q)$ .

*Proof.* Since  $\ell(t, p) < \ell(t, q)$ ,  $MF$  processes  $p$  earlier than  $q$ , which means that  $\text{arr}(p) \leq \text{arr}(q)$ . Thus, in the case of  $x = 1$ , if  $\text{arr}(p) \leq \text{arr}(q)$ , then  $g(p) \leq g(q)$  by the contrapositive of Lemma 3.6. In the same way, using the contrapositive of Lemma 3.6,  $g(p) \leq g(q)$  if  $x \neq 1$  and  $\text{arr}(p) < \text{arr}(q)$ . In the case where both  $x \neq 1$  and  $\text{arr}(p) = \text{arr}(q)$ ,  $MF$  processes a packet with a smaller block number earlier by definition, and hence  $g(p) \leq g(q)$ .  $\square$

Now we are ready to prove Lemma 3.1. When  $M = 1$ , clearly  $a_1 = 1$  because  $M \in G$  by definition. When  $M \geq 2$ , we show that at any moment there are at least  $\lfloor A/2 \rfloor$  frames  $f$  with block number 1 such that  $f$  is valid for  $MF$ .  $MF$  accepts at least  $\lfloor A/2 \rfloor$  1-packets with block number 1 according to Lemma 3.5(b). If  $MF$  does not preempt any packet with block number 1, the statement is clearly true. Then, suppose that at an event time  $t$ ,  $MF$  preempts an  $x(\in [2, k])$ -packet with block number 1. By Case 2.2.2 in  $MF$ ,  $MF$  stores  $A$   $x$ -packets in its buffer at  $t-$ . Moreover, all the  $x$ -packets in  $MF$ 's buffer are queued in ascending order of block number by Lemma 3.7. Thus, for each  $x$ -packet  $p$  such that  $\ell(t+, p) \in [1, \lfloor A/2 \rfloor]$ ,  $g(p) = 1$ . As a result,  $h_{MF,1}(t+) \geq \lfloor A/2 \rfloor$ , which proves the lemma.  $\square$

### 3.3.3 Proof of Lemma 3.2

In order to prove Lemma 3.2, we bound the number of 1-packets accepted by  $OPT$  during a time interval by the number of 1-packets accepted by  $GR_1$ .

**Lemma 3.8** *Let  $t_1$  and  $t_2(> t_1)$  be any non-event times. Suppose that  $GR_1$  accepts  $w(\geq 1)$  1-packets during time  $[t_1, t_2]$ , and let  $p$  be the first 1-packet accepted by  $GR_1$  during time  $[t_1, t_2]$ . Then, the number of 1-packets accepted by  $OPT$  during time  $[t_p-, t_2]$  is at most  $w + B - 1$ . Moreover, when  $t_1$  is a time before the beginning of the input, the number of 1-packets accepted by  $OPT$  during time  $[t_p-, t_2]$  is at most  $w$ .*

*Proof.* Define  $OPT_1$  as the offline algorithm that accepts exactly the same 1-packets as accepted by  $OPT$  and ignores  $j(\in [2, k])$ -packets. Let  $x$  (respectively  $x'$ ) be the number of 1-packets accepted by  $GR_1$  but not accepted by  $OPT_1$  (respectively accepted by  $OPT_1$  but not accepted by  $GR_1$ ) during time  $[t_p-, t_2]$ . Also, let  $x''$  be the number of 1-packets accepted by both  $GR_1$  and  $OPT_1$  during time  $[t_p-, t_2]$ . Since  $GR_1$  accepts  $w$  packets during time  $[t_p-, t_2]$ ,  $x + x'' = w$ . In what follows, we bound  $x' + x''$  from above.

For a non-event time  $t$  and an algorithm  $ALG'(\in \{OPT_1, GR_1\})$ , let  $f_{ALG'}(t)$  denote the number of 1-packets in  $ALG'$ 's buffer at  $t$ . Since  $GR_1$  accepts 1-packets greedily and  $OPT_1$  accepts only 1-packets,  $f_{GR_1}(t) - f_{OPT_1}(t) \geq 0$  holds for any  $t$ . Let  $y$  (respectively  $y'$ ) denote the number of 1-packets transmitted by  $GR_1$  (respectively  $OPT_1$ ) during time  $[t_p-, t_2]$ . Since  $f_{GR_1}(t) - f_{OPT_1}(t) \geq 0$  for any  $t$ ,  $GR_1$  transmits a 1-packet whenever  $OPT_1$  does so, and hence  $y \geq y'$ . By an easy calculation,  $f_{GR_1}(t_2) = f_{GR_1}(t_p-) + x + x'' - y$  and  $f_{OPT_1}(t_2) = f_{OPT_1}(t_p-) + x' + x'' - y'$ . By the above equalities and inequalities,

$$\begin{aligned} 0 &\leq f_{GR_1}(t_2) - f_{OPT_1}(t_2) = f_{GR_1}(t_p-) + x + x'' - y - (f_{OPT_1}(t_p-) + x' + x'' - y') \\ &= f_{GR_1}(t_p-) - f_{OPT_1}(t_p-) + x - x' - y + y' \leq f_{GR_1}(t_p-) - f_{OPT_1}(t_p-) + x - x'. \end{aligned}$$

That is,  $x' \leq f_{GR_1}(t_p-) - f_{OPT_1}(t_p-) + x$ . Hence,  $x' + x'' \leq f_{GR_1}(t_p-) - f_{OPT_1}(t_p-) + x + x'' = f_{GR_1}(t_p-) - f_{OPT_1}(t_p-) + w$  holds. Furthermore,  $f_{GR_1}(t_p-) - f_{OPT_1}(t_p-) \leq B - 1$  since  $GR_1$  accepts  $p$ , namely,  $GR_1$ 's buffer is not full just before the decision time of  $p$ . Thus,  $x' + x'' \leq B - 1 + w$ .

Finally we consider the case where  $t_1$  is a time before the beginning of the input. Since  $f_{GR_1}(t_p-) = f_{OPT_1}(t_p-) = 0$ ,  $x' + x'' \leq f_{GR_1}(t_p-) - f_{OPT_1}(t_p-) + w = w$  holds.  $\square$

We are ready to give the proof of Lemma 3.2. The proof of (a) is almost the same as that of Lemma 3.1. By the assumption that  $c = 0$  or  $c \in [\lfloor A/2 \rfloor, 3B - 1]$ ,  $MF$  accepts at least  $\lfloor A/2 \rfloor$  1-packets with block number  $M$  according to Lemma 3.5(b). If  $MF$  does not preempt any packet

with block number  $M$ , the statement is clearly true. Then, suppose that at an event time  $t$ ,  $MF$  preempts an  $x(\in [2, k])$ -packet with block number  $M$ . By Case 2.2.2,  $MF$  stores  $A$   $x$ -packets in its buffer at  $t-$ . Moreover, all the  $x$ -packets in  $MF$ 's buffer are queued in ascending order by their block numbers by Lemma 3.7. Thus, for each  $x$ -packet  $p$  such that  $\ell(t+, p) \in [\lfloor A/2 \rfloor + 1, A]$ ,  $g(p) = M$ . As a result,  $h_{MF, M}(t+) \geq \lfloor A/2 \rfloor$ , which proves the part (a) of Lemma 3.2.

As for (b), since  $c \in [1, \lfloor A/2 \rfloor - 1]$  by the assumption of (b), all the 1-packets with block number  $M$  which are accepted by  $MF$  are the same as those of  $GR_1$ . Then, let  $p'$  ( $p''$ ) be the first ( $h_{MF, M}(\tau)$ th) 1-packet accepted by  $MF$  whose block number is  $M$ . If we set  $t_1 = t_{p'} -$  and  $t_2 = t_{p''} +$  in Lemma 3.8, then  $w = h_{MF, M}(\tau)$ . Thus we have that  $h_{MF, M}(\tau) + B - 1 \geq h_{OPT, M}(\tau)$ . Part (c) can be proved in a similar way to (b), by applying the latter part of Lemma 3.8.  $\square$

### 3.3.4 Proof of Lemma 3.3

To prove Lemma 3.3, we first show the next three lemmas. We show in Lemma 3.9 that for non-consecutive two good block numbers  $a_j$  and  $a_{j+1}$ , there must be a moment when an  $x(\in [2, k])$ -packet with block number  $a_j$  and an  $x$ -packet with block number  $a_{j+1}$  exist in  $MF$ 's buffer simultaneously. This is a consequence of using middle-drop policy. We then show in Lemma 3.11 that in such a case, the number of packets accepted by  $OPT$  with block number  $a_j, a_{j+1}, \dots, a_{j+1} - 1$  can be bounded.

**Lemma 3.9** *Suppose that  $a_{j+1} - a_j \geq 2$  for an integer  $j(\in [1, m - 1])$ . Then there exist two  $x$ -packets  $q$  and  $q'$  for some integer  $x \in [2, k]$  such that  $g(q) = a_j$ ,  $g(q') = a_{j+1}$ , and both  $q$  and  $q'$  are stored in  $MF$ 's buffer at the same time.*

*Proof.* For a non-event time  $t$ , we say that a block number  $u$  is *good* at  $t$  if  $u = M$  or at least  $\lfloor A/2 \rfloor$  frames with the block number  $u$  are valid at  $t$ , and *bad* at  $t$  otherwise. Note that the set of good block numbers at the end of the input coincides the set  $G$  (see Sec. 3.2 for the definition of  $G$ ). Since  $a_{j+1} - a_j \geq 2$ , there must be at least one block number between  $a_j$  and  $a_{j+1}$ . Those block numbers were initially good but turned bad at some event time, since  $a_j$  and  $a_{j+1}$  are good block numbers that are consecutive at the end of the input. Let  $u$  ( $a_j < u < a_{j+1}$ ) be the block number that turned bad lastly among them. The event time when block number  $u$  turns bad is the decision time  $t_{p'}$  when some  $x(\in [2, k])$ -packet  $p'$  arrives. Specifically,  $MF$  accepts  $p'$  at  $t_{p'}$ , and preempts an  $x$ -packet  $p''$  with block number  $u(= g(p''))$  at Case 2.2.2 such that  $\ell(t_{p'}-, p'') = \lfloor A/2 \rfloor + 1$ . Moreover,  $MF$  preempts all the packets with block number  $u$  in  $MF$ 's buffer by executing Case 2.2.2.1.

Now we discuss the block numbers of packets in  $MF$ 's buffer before or after  $t_{p'}$ . By the definition of Case 2.2.2 in  $MF$ , the number of  $x$ -packets in  $MF$ 's buffer at  $t_{p'} -$  is  $A$ , and among them, exactly  $\lfloor A/2 \rfloor$  ones are of block number  $g(p'')$  (or  $\lfloor A/2 \rfloor - 1$  ones excluding  $p''$ ). In addition, all the  $x$ -packets in  $MF$ 's buffer are queued in ascending order by their block numbers by Lemma 3.7. Hence, (a)  $g(p'') > g(p)$  holds, where  $p$  is the  $x$ -packet such that  $\ell(t_{p'}-, p) = \ell(t_{p'}+, p) = 1$ . Also,  $MF$  accepts  $p'$ , and preempts all the packets with block number  $g(p'')$  at Case 2.2.2.1. Thus,  $g(p') \neq g(p'')$ , which means that (b)  $g(p'') < g(p')$  holds according to Lemma 3.7.

Now if  $a_j < g(p)$ , then  $a_j < g(p) < g(p'')$  by (a). This contradicts the definition of  $u$  since there still remains a good block number  $g(p)$  between  $a_j$  and  $u(= g(p''))$ . Hence  $a_j \geq g(p)$ . In the same way, if  $a_{j+1} > g(p')$ , then  $g(p'') < g(p') < a_{j+1}$  by (b). We have the contradiction as well, which means that (c)  $a_{j+1} \leq g(p')$ .

In the following, we prove that  $q$  and  $q'$  mentioned in this lemma exist in the buffer at time  $t_{p'}+$ . We first show the existence of  $q$ . Let us consider the case of  $a_j = g(p)$ . In this case,  $p$  is clearly stored in  $MF$ 's buffer at  $t_{p'}+$ , and  $p$  satisfies the condition of  $q$ . Next, we consider the case of  $a_j > g(p)$ . Since  $a_j$  is a good block number by definition, there must be a packet  $p'''$  such that  $a_j = g(p''')$  and  $p'''$  is valid at  $t_{p'}+$ . Then,  $g(p) < g(p''') = a_j < a_{j+1} \leq g(p')$  by (c) and hence  $\text{arr}(p) \leq \text{arr}(p''') \leq \text{arr}(p')$  by Lemma 3.6. Note that  $MF$  stores both  $p'$  and  $p$  in its buffer at  $t_{p'}+$ , and  $p'''$  is valid at  $t_{p'}+$  by the above definition. Therefore,  $p'''$  is stored in  $MF$ 's buffer at  $t_{p'}+$ , and thus this  $p'''$  satisfies the condition of  $q$ . The case of  $q'$  can be proven in the same way as  $q$ . Namely, if  $a_{j+1} = g(p')$ , then let  $q' = p'$ . Also, if  $a_{j+1} < g(p')$ , then there must be  $q'$  satisfying  $g(p) \leq a_j < a_{j+1} = g(q') < g(p')$ . This completes the proof.  $\square$

**Lemma 3.10** *For any non-event time  $t$  and  $x \in [2, k]$ , let  $p$  be an  $x$ -packet valid for  $MF$  at  $t$ . Then the number of  $x$ -packets  $q$  such that  $OPT$  accepts  $q$ ,  $\text{arr}(p) < \text{arr}(q)$ , and  $g(q) \in [1, g(p) - 1]$  is at most  $B$ .*

*Proof.* Let  $p_1$  be the 1-packet corresponding to  $p$ ,  $q'$  be an  $x$ -packet accepted by  $OPT$ , and  $q'_1$  be the 1-packet corresponding to  $q'$ . As we assume that  $OPT$  never accepts a packet of an incompleting frame,  $q'_1$  is accepted by  $OPT$ . Since the input is order-respecting,  $\text{arr}(p) \geq \text{arr}(q')$  if  $\text{arr}(p_1) > \text{arr}(q'_1)$ , that is, such  $q'$  does not satisfy the second condition of  $q$  in the statement of this lemma. Since the block numbers of 1-packets are monotonically non-decreasing in an arrival order,  $g(p_1) \leq g(q'_1)$  if  $\text{arr}(p_1) < \text{arr}(q'_1)$ , namely, such  $q'$  does not satisfy the third condition of  $q$ . Thus, only  $q'$  such that  $\text{arr}(p_1) = \text{arr}(q'_1)$  can satisfy all the conditions of  $q$ . Since the buffer size is  $B$ , the number of such  $q'_1$  accepted by  $OPT$  is at most  $B$ , which completes the proof.  $\square$

**Lemma 3.11** *Let  $p$  and  $p'$  be  $x(\in [2, k])$ -packets stored in  $MF$ 's buffer at the same time, and suppose that  $g(p') - g(p) \geq 2$ . Then, if  $g(p) \geq 2$ , the number of  $x$ -packets  $\tilde{p}$  such that  $g(\tilde{p}) \in [g(p), g(p') - 1]$ , and  $\tilde{p}$  is accepted by  $OPT$  is at most  $5B + A - 4$ . Moreover, if  $g(p) = 1$ , the number of  $x$ -packets  $\hat{p}$  such that  $g(\hat{p}) \in [1, g(p') - 1]$ , and  $\hat{p}$  is accepted by  $OPT$  is at most  $4B + A - 3$ .*

*Proof.* First, we consider the case of  $g(p) \geq 2$ . Let  $q$  be an  $x$ -packet satisfying the conditions of the lemma, i.e., an  $x$ -packet  $q$  such that  $g(q) \in [g(p), g(p') - 1]$  and  $q$  is accepted by  $OPT$ . Note that  $\text{arr}(p) \leq \text{arr}(p')$  by Lemma 3.6 (b) because  $g(p) < g(p')$ . We count the number of such  $q$  for each of the cases (i)  $\text{arr}(q) < \text{arr}(p)$ , (ii)  $\text{arr}(p) \leq \text{arr}(q) \leq \text{arr}(p')$ , and (iii)  $\text{arr}(p') < \text{arr}(q)$ .

(i) First, note that there is no  $q$  such that  $g(q) \in [g(p) + 1, g(p') - 1]$  by Lemma 3.6, since  $\text{arr}(q) < \text{arr}(p)$ . Hence, we focus on  $q$  such that  $g(q) = g(p)$ . Let  $p_1$  and  $q_1$  be the 1-packets corresponding to  $p$  and  $q$ , respectively, and suppose that  $p_1$  ( $q_1$ ) is the  $j$ th (first) 1-packet accepted by  $MF$  with block number  $g(p)$ . To count the number of  $q$  satisfying the condition, we count the number of corresponding  $q_1$ . Note that  $g(q_1) = g(p_1)$  since  $g(q) = g(p)$ . By the definition of  $MF$ , the  $j(\in [1, A])$ th 1-packet accepted by  $MF$  is also accepted by  $GR_1$ . If we set  $t_1 = t_{p'_1} -$  and  $t_2 = t_{p_1} -$ , then  $w = j - 1$  in Lemma 3.8, and Lemma 3.8 implies that the number of  $q_1$  such that  $\text{arr}(q_1) < \text{arr}(p_1)$  is at most  $j - 1 + B - 1$ . This is at most  $A + B - 2$  since  $j \leq A$  by Lemma 3.5(b). The number of  $q_1$  such that  $\text{arr}(q_1) = \text{arr}(p_1)$  is at most  $B$ , since the buffer size is  $B$ . Finally, the number of  $q_1$  such that  $\text{arr}(q_1) > \text{arr}(p_1)$  is zero by the order-respecting assumption because  $\text{arr}(q) < \text{arr}(p)$ . Hence, the number of  $q$  in Case (i) is at most  $(A + B - 2) + B = 2B + A - 2$ .

(ii) Let  $t$  be any non-event time when both  $p$  and  $p'$  are stored in  $MF$ 's buffer. Let  $w' = \text{arr}(p)$  and suppose that the delivery subphase just before  $t$  is in the  $w''$ th phase. Then, the number of delivery subphases during  $[w', w'']$  is  $w'' - w' + 1$ . Since  $p$  is still stored in  $MF$ 's buffer at  $t$ ,

$w'' - w' + 1 \leq B - 1$  (as otherwise,  $MF$  must have transmitted  $p$  before  $t$ ). The number of  $x$ -packets which arrive during  $[w', w'']$  and are accepted by  $OPT$  is at most  $B + w'' - w' \leq 2B - 2$  by a similar argument to the proof of Lemma 3.5(a). Thus, the number of  $x$ -packets  $q$  in this case is at most  $2B - 2$ .

(iii) By Lemma 3.10, the number of  $x$ -packets  $q$  in this case is at most  $B$ .

Putting (i), (ii), and (iii) together, the number of  $x$ -packets  $q$  is at most  $(2B + A - 2) + (2B - 2) + B = 5B + A - 4$ .

For  $g(p) = 1$ , the argument is the same as the case of  $g(p) \geq 2$ , except that at an application of Lemma 3.8 in Case (i), we let  $t_1$  be the time before the beginning of the input. Then, the number of  $q_1$  such that  $q_1$  is accepted by  $OPT$ ,  $g(q_1) = g(p_1)$ , and  $\text{arr}(q_1) < \text{arr}(p_1)$  is at most  $A - 1$ , instead of  $A + B - 2$  in the case of  $g(p) \geq 2$ . Then the number of  $x$ -packets  $q$  in question is at most  $(B + A - 1) + (2B - 2) + B = 4B + A - 3$ .  $\square$

Now we are ready to give the proof of Lemma 3.3. Fix the block number  $u (\neq M)$ . We count the number of 1-packets  $p$  accepted by  $OPT$  such that  $g(p) = u$ . Note that the number of 1-packets with block number  $u$  accepted by  $GR_1$  is  $3B$ . Let  $q$  ( $q'$ ) be the first (last, i.e.,  $3B$ th) 1-packet accepted by  $GR_1$  with block number  $u$ . Also, let  $q''$  be the first 1-packet accepted by  $GR_1$  after  $t_{q'}$ . Then  $q''$  has the block number  $u + 1$  by definition, and hence any packet with block number  $u$  arrives during time  $[t_q, t_{q''}]$ . By applying Lemma 3.8 with  $t_1 = t_q$  and  $t_2 = t_{q''}$ , i.e.,  $w = 3B$ , the number of 1-packets  $p$  accepted by  $OPT$  such that  $g(p) = u$  is at most  $3B + B - 1 = 4B - 1$ . When  $u = M$ , the same upper bound can be obtained by almost the same argument as the above. We use this fact several times in the following.

(a) By the above discussion, the number of 1-packets  $p$  accepted by  $OPT$  such that  $g(p) = a_m$  is at most  $4B - 1$ , and hence the number of frames  $f$  completed by  $OPT$  such that  $g(f) = a_m$  is at most  $4B - 1$ .

(b) In the case of  $a_2 = a_1 + 1$ , by the same argument as (a) we can conclude that the number of completed frames is at most  $4B - 1 \leq 4B + A - 3$ . If  $a_2 \geq a_1 + 2$ , we know by Lemma 3.9 that two  $x$ -packets  $\hat{p}$  and  $\tilde{p}$  such that  $g(\hat{p}) = a_1$  and  $g(\tilde{p}) = a_2$  are stored in  $MF$ 's buffer at the same time. Then by Lemma 3.11, the number of  $x$ -packets  $p$  accepted by  $OPT$  such that  $g(p) \in [a_1, a_2 - 1]$  is at most  $4B + A - 3$  (recall that  $a_1 = 1$ ). By the same argument as above, we can conclude that the number of frames completed by  $OPT$  such that  $g(f) \in [a_1, a_2 - 1]$  is also at most this number.

(c) The argument is almost the same as (b). The only difference is that here we use the fact that for  $i \geq 2$ , the number of  $x$ -packets  $p$  accepted by  $OPT$  such that  $g(p) \in [a_i, a_{i+1} - 1]$  is at most  $5B + A - 4$ , which is due to Lemma 3.11.  $\square$

## 4 Lower Bound for Deterministic Algorithms

In this section, we show a lower bound for deterministic algorithms.

**Theorem 4.1** *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  $B \leq k - 2$ .*

*Proof.* Fix an online algorithm  $ALG$ . Let us consider the following input  $\sigma$ . (See Figure 1.) At the 0th phase,  $2B$  1-packets arrive.  $ALG$  accepts at most  $B$  1-packets, and  $OPT$  accepts  $B$  1-packets that are not accepted by  $ALG$ . Let  $C$  ( $D$ , respectively) be the set of the 1-packets accepted by

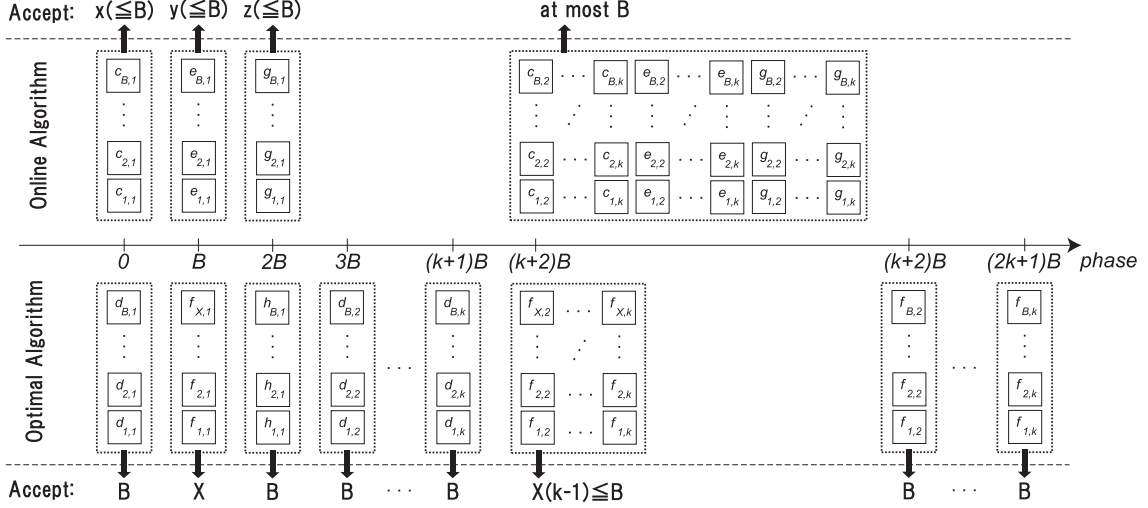


Figure 1: Lower Bound Instance. Each square denotes an arriving packet accepted by an online algorithm or  $OPT$ . In the figure  $X = \lfloor \frac{B}{k-1} \rfloor$ .

$ALG$  ( $OPT$ , respectively). At the  $i$ th phase ( $i \in [1, B-1]$ ), no packets arrive. Hence, just after the  $(B-1)$ st phase, both  $ALG$ 's and  $OPT$ 's queues are empty (since  $B$  delivery subphases occur).

At the  $B$ th phase,  $B + \lfloor \frac{B}{k-1} \rfloor$  1-packets arrive in the same manner as the first  $2B$  1-packets.  $ALG$  can accept at most  $B$  1-packets, and  $OPT$  accepts  $\lfloor \frac{B}{k-1} \rfloor$  1-packets that are not accepted by  $ALG$ . Let  $E$  ( $F$ , respectively) be the set of the packets accepted by  $ALG$  ( $OPT$ , respectively). At the  $i$ th phase ( $i \in [B+1, 2B-1]$ ), no packets arrive, and both  $ALG$ 's and  $OPT$ 's queues are empty just after the  $(2B-1)$ st phase.

Once again at the  $2B$ th phase,  $2B$  1-packets arrive.  $ALG$  accepts at most  $B$  1-packets, and  $OPT$  accepts  $B$  1-packets that are not accepted by  $ALG$ . Let  $G$  ( $H$ , respectively) be the set of the 1-packets accepted by  $ALG$  ( $OPT$ , respectively). This is the end of the arrivals and deliveries of 1-packets. At the  $i$ th phase ( $i \in [2B+1, 3B-1]$ ), no packets arrive, and hence just before the  $3B$ th phase, both  $ALG$ 's and  $OPT$ 's queues are empty.

For each  $j = 2, \dots, k$ , the  $B$   $j$ -packets corresponding to 1-packets in  $D$  arrive at the  $(j+1)B$ th phase.  $OPT$  accepts and transmits them. (There is no incentive for  $ALG$  to accept them.) Next, all the packets corresponding to all the 1-packets in  $C \cup E \cup F \cup G$  arrive at the  $(k+2)B$ th phase. Since  $ALG$  needs to accept all the  $k-1$  packets of the same frame to complete it, the number of frames  $ALG$  can complete is at most  $\lfloor \frac{B}{k-1} \rfloor$ .  $OPT$  accepts all the  $\lfloor \frac{B}{k-1} \rfloor (k-1)$  packets corresponding to all the 1-packets in  $F$ . Note that this is possible because  $\lfloor \frac{B}{k-1} \rfloor (k-1) \leq B$ . Hence,  $OPT$  completes all the  $\lfloor \frac{B}{k-1} \rfloor$  frames of  $F$ .

After which all the packets corresponding to 1-packets in  $H$  arrive one after the other, and  $OPT$  can accept and transmit them. Note that the input sequence is order-respecting.

By the above argument, we have  $V_{ALG}(\sigma) \leq \lfloor \frac{B}{k-1} \rfloor$  and  $V_{OPT}(\sigma) = 2B + \lfloor \frac{B}{k-1} \rfloor$ . Therefore, if  $B \geq k-1$ ,  $\frac{V_{OPT}(\sigma)}{V_{ALG}(\sigma)} \geq \frac{2B}{\lfloor \frac{B}{k-1} \rfloor} + 1$ . If  $B \leq k-2$ , the competitive ratio of  $ALG$  is unbounded.  $\square$

## 5 Lower Bound for Randomized Algorithms

**Theorem 5.1** *When  $k \geq 3$ , the competitive ratio of any randomized algorithm is at least  $k - 1 - \epsilon$  for any constant  $\epsilon$  against an oblivious adversary.*

*Proof.* Fix an arbitrary randomized online algorithm  $ALG$ . Let  $y$  be a large integer that will be fixed later. Our adversarial input  $\sigma$  consists of  $(k - 1)yB$  frames. These frames are divided into  $k - 1$  groups each with  $yB$  frames. Also, frames of each group are divided into  $y$  subgroups each with  $B$  frames. For each  $i \in [1, k - 1]$  and  $j \in [1, y]$ , let  $F(i, j)$  be the set of frames in the  $j$ th subgroup of the  $i$ th group and let  $F(i) = \cup_j F(i, j)$ . For each  $x \in [1, k]$ , let  $P(i, j, x)$  be the set of  $x$ -packets of the frames in  $F(i, j)$  and let  $P(i, x) = \cup_j P(i, j, x)$ .

We first give a very rough idea of how to construct the adversary. Among the  $k - 1$  groups defined above, one of them is a good group. In the first half of the input (from phase 0 to phase  $(k - 1)yB - 1$ ), the adversary gives packets to the online algorithm in such a way that the algorithm cannot distinguish the good group. Also, since the buffer size is bounded, the algorithm must give up many frames during the first half; only  $yB$  frames can survive at the end of the first half. In the second half of the input, remaining packets are given in such a way that  $k$ -packets from the bad groups arrive at a burst, while  $k$ -packets from the good group arrive one by one. Hence, if the algorithm is lucky enough to keep many packets of the good group (say, Group 1) at the end of the first half, then it can complete many frames eventually. However, such an algorithm behaves very poorly for an input in which Group 1 is bad. Therefore, the best strategy of an online algorithm (even randomized one) is to keep equal number of frames from each group during the first half.

Before showing our adversarial input, we define a subsequence of an input. For any  $t$ , suppose that  $B$  packets of  $P(i, j, x)$  arrive at the  $t$ th phase and no packets arrive during  $t + 1$  through  $(t + B - 1)$ st phases. Let us call this subsequence a *subround of  $P(i, j, x)$  starting at the  $t$ th phase*. Notice that if we focus on a single subround, an algorithm can accept and transmit all the packets of  $P(i, j, x)$  by the end of the subround. A *round of  $P(i, x)$  starting at the  $t$ th phase* is a concatenation of  $y$  subrounds of  $P(i, j, x)$  ( $j \in [1, y]$ ), where each subround of  $P(i, j, x)$  starts at the  $(t + (j - 1)B)$ th phase. (See the left figure in Fig. 2.)

Our input consists of rounds of  $P(i, x)$  starting at the  $(i + x - 2)yB$ th phase, for  $i \in [1, k - 1]$  and  $x \in [1, k - 1]$ . (See Fig. 3.) Note that any two rounds  $P(i, x)$  and  $P(i', x')$  start simultaneously if  $i + x = i' + x'$ . Currently, the specification of the arrival of packets in  $P(i, x)$  for  $x = k$  is missing. This is the key for the construction of our adversary and will be explained shortly.

Consider  $k - 1$  rounds (of  $P(1, k - 1), P(2, k - 2), \dots, P(k - 1, 1)$ ) starting at the  $(k - 2)yB$ th phase, which occur simultaneously. Note that for each  $j$ , at the  $j$ th subround of these  $k - 1$  rounds,  $ALG$  can accept at most  $B$  packets (out of  $(k - 1)B$  ones) because of the size constraint of the buffer. For each  $j \in [1, y]$ , let  $A_{i,j}$  denote the expected number of packets that  $ALG$  accepts from  $P(i, j, k - i)$ . By the above argument, we have that  $\sum_i A_{i,j} \leq B$  and hence  $\sum_i \sum_j A_{i,j} \leq yB$ . Let  $A_i = \sum_j A_{i,j}$  and let  $A_z$  be the minimum among  $A_1, A_2, \dots, A_{k-1}$  (ties are broken arbitrarily). Note that  $A_z \leq \frac{yB}{k-1}$  since  $\sum_i A_i = \sum_i \sum_j A_{i,j} \leq yB$ . Also, note that since  $A_i$  is an expectation,  $z$  is determined only by the description of  $ALG$  (and not by the actual behavior of  $A$ ).

We now explain the arrival of packets in  $P(i, k)$  ( $i \in [1, k - 1]$ ). (See the right figure in Fig. 2.) For  $i \neq z$ , all the  $yB$  packets in  $P(i, k)$  arrive simultaneously at the  $(i + k - 2)yB$ th phase. As for  $i = z$ , packets are given as a usual round, i.e., we have a round of  $P(z, k)$  starting at  $(z + k - 2)yB$ . It is not hard to verify that this input is order-respecting. Also, it can be easily verified that our adversary is oblivious because the construction of the input does not depend on the actual behavior

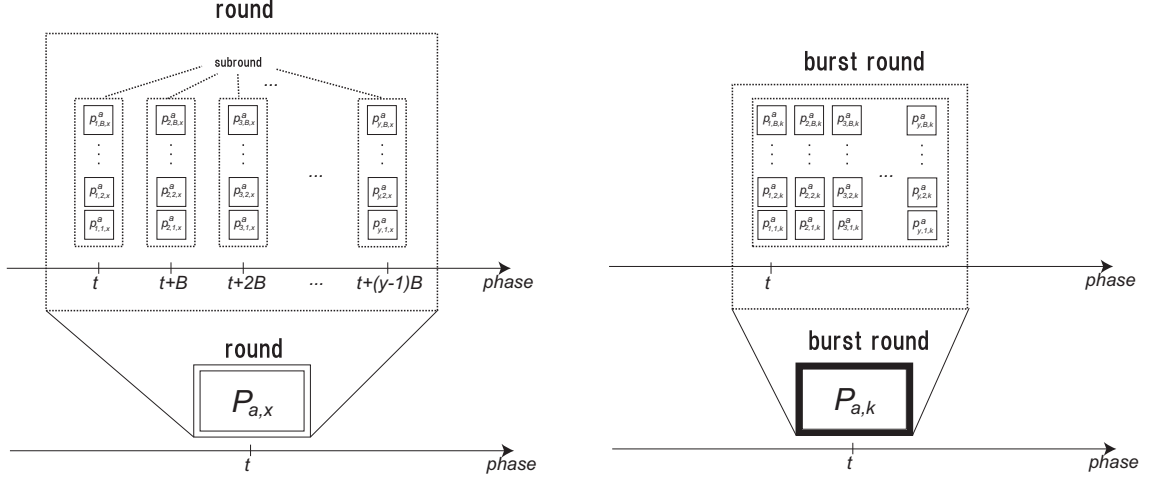


Figure 2:  $P(a, x)$  is written as  $P_{a,x}$  in this figure. The left figure shows a round of  $P(a, x)$  except for the case where  $a \neq z$  and  $x = k$ . On the other hand, the right figure shows a round of  $P(a, k)$  for each  $a \in [1, k-1]$  such that  $a \neq z$ .

of  $ALG$ . Specifically,  $z$  depends on only the values of  $A_{i,j}$  ( $i \in [1, k-1], j \in [1, y]$ ), and  $\sigma$  can be constructed not with time but in advance.

First, note that  $OPT$  can accept and transmit all the packets in  $P(z, x)$  for any  $x$ . Therefore,  $OPT$  can complete all the  $yB$  frames in  $F(z)$  and hence  $V_{OPT}(\sigma) \geq yB$ . On the other hand, since all the packets in  $P(i, k)$  ( $i \neq z$ ) arrive simultaneously,  $ALG$  can accept at most  $B$  packets of them and hence can complete at most  $B$  frames of  $F(i)$  for each  $i$ . As for  $F(z)$ ,  $ALG$  can complete at most  $A_z \leq \frac{yB}{k-1}$  frames of them and hence  $\mathbb{E}[V_{ALG}(\sigma)] \leq \frac{yB}{k-1} + (k-2)B$ . If we take  $y \geq \frac{(k-1)^2(k-2)}{\epsilon} - (k-1)(k-2)$ , we have that

$$\frac{V_{OPT}(\sigma)}{\mathbb{E}[V_{ALG}(\sigma)]} \geq \frac{yB}{(yB)/(k-1) + (k-2)B} = k-1 - \frac{(k-1)^2(k-2)}{y + (k-1)(k-2)} \geq k-1 - \epsilon.$$

□

## 6 Conclusion

In this paper, we have improved an upper bound on the competitive ratio for  $k$ -OFTM, showing the  $\Theta(k)$ -competitiveness of the problem when  $B \geq 2k$ . We also have presented lower bounds for deterministic and randomized settings. When  $k-1 \leq B < 2k$ , our proof for the upper bound does not work because there exists a  $j$ -subbuffer whose size is one and hence it is impossible for two  $j$ -packets to exist in the buffer simultaneously. Developing competitive algorithms for this case is one of interesting future work. Moreover,  $MF$  is a preemptive algorithm whereas the  $O(k^2)$ -competitive algorithm  $SP$  in [25] is non-preemptive. Therefore it is interesting to design (or prove the non-existence of) a non-preemptive algorithm whose competitive ratio is  $O(k)$ .

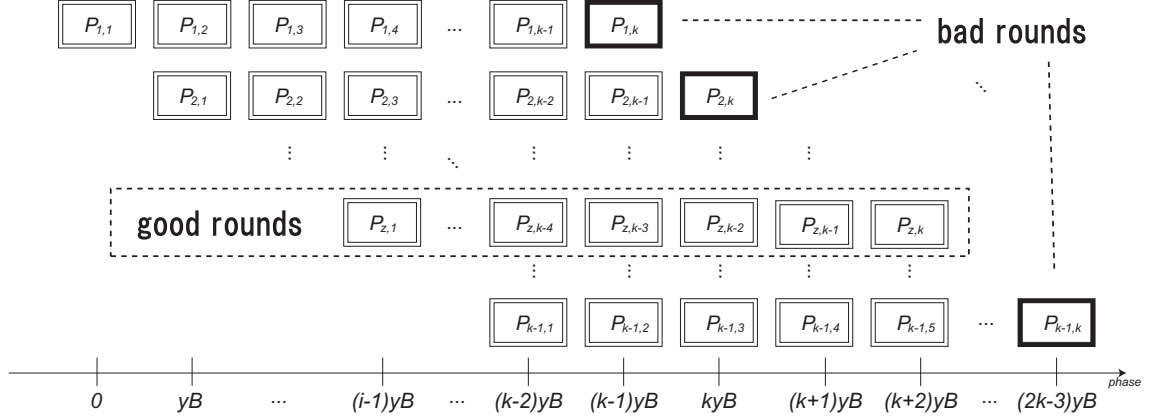


Figure 3: Lower bound instance for randomized algorithms

## 7 Acknowledgments

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

## References

- [1] W. Aiello, Y. Mansour, S. Rajagopalan, 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,” *Distributed Computing*, Vol. 2848, 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. Litichevsky, “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,” *Algorithmica*, Vol.68, No.2, pp, 426–447, 2014.
- [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,” *Algorithmica*, Vol.53, No.4, pp. 523–548, 2009.
- [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. M. Halldórsson, B. Patt-Shamir and D. Rawitz, “Online Scheduling with Interval Conflicts,” *Theory of Computing Systems*, Vol.53, No.2, pp. 300–317, 2013.
- [16] J. Kawahara, and K. M. Kobayashi, “Optimal Buffer Management for 2-Frame Throughput Maximization,” *Computer Networks*, Vol. 91, pp. 804–820, 2015.
- [17] 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.
- [18] 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.
- [19] A. Kesselman, Y. Mansour and R. van Stee, “Improved competitive guarantees for QoS buffering,” *Algorithmica*, Vol.43, No.1-2, pp. 63–80, 2005.
- [20] A. Kesselman and A. Rosén, “Scheduling policies for CIOQ switches,” *Journal of Algorithms*, Vol. 60, No. 1, pp. 60–83, 2006.
- [21] 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.
- [22] A. Kesselman, K. Kogan and M. Segal, “Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing,” *Distributed Computing*, Vol.23, No.3, pp. 163–175, 2010.
- [23] A. Kesselman, K. Kogan and M. Segal, “Best effort and priority queuing policies for buffered crossbar switches,” *Chicago Journal of Theoretical Science*, pp. 1–14, 2012.
- [24] A. Kesselman, K. Kogan and M. Segal, “Improved competitive performance bounds for CIOQ switches,” *Algorithmica*, Vol.63, No.1-2, pp. 411–424, 2012.
- [25] A. Kesselman, B. Patt-Shamir and G. Scalosub, “Competitive buffer management with packet dependencies,” *Theoretical Computer Science*, Vol.489–490, pp. 75–87, 2013.
- [26] 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.

- [27] 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.
- [28] 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.
- [29] Y. Mansour, B. Patt-Shamir, and D. Rawitz, “Competitive router scheduling with structured data,” *Theoretical Computer Science*, Vol. 530, pp. 12–22, 2014.
- [30] G. Scalosub, P. Marbach and J. Liebeherr, “Buffer management for aggregated streaming data with packet dependencies,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 3, pp. 439–449, 2013.
- [31] D. D. Sleator, and R. E. Tarjan, “Amortized efficiency of list update and paging rules,” *Communications of the ACM*, Vol. 28, No. 2, pp. 202–208, 1985.
- [32] M. Sviridenko, “A lower bound for on-line algorithms in the FIFO model,” unpublished manuscript, 2001.

## A Lower bound for SP

We give an input  $\sigma$  for which  $SP$ ’s competitive ratio is as bad as  $\Omega(k^2)$ . For ease of presentation, let  $D = 3B$  and  $N = 3 \times 2^{k-1}$ .  $\sigma$  consists of  $NB$  frames  $f_1, \dots, f_{NB}$ . For any  $i \in [1, NB]$  and any  $j \in [1, k]$ , let  $p_{i,j}$  denote the  $j$ -packet of  $f_i$ . Fig. 4 shows a pseudocode of generating  $\sigma$ . Note that in  $\sigma$ , all the 1-packets arrive first. After that, all the 2-packets arrive, then all the 3-packets do, and so on. An example of  $\sigma$  for  $k = 5$  is depicted in Figs. 5 through 9, corresponding to 1- through 5-packets, respectively. Each figure consists of two graphs. An upper graph shows the arrival phase of each packet, where the horizontal axis shows the packet index and vertical axis shows the phase. For example, Fig. 5 shows that 1-packets  $p_{i,1}$  ( $i \in [1, B]$ ) arrive at the 0th phase, indicated as (1), 1-packets  $p_{i,1}$  ( $i \in [B + 1, 2B]$ ) arrive at the  $B$ th phase, indicated as (2), and so on. (We assume that Figs. 5 through 9 are printed in color. Please refer to the PDF version if necessary.)

First, consider  $SP$ ’s behavior. Without loss of generality, we assume that  $SP$  prioritizes frames with smaller indices, e.g., if two packets  $p_{i,j}$  and  $p_{i',j}$  with  $i < i'$  arrive at the same time and  $SP$  is able to accept only one packet, then  $SP$  accepts  $p_{i,j}$  and rejects  $p_{i',j}$ . The lower graphs of Figs. 5 through 9 show the behaviors of  $SP$  and  $OPT$ . For example, Fig. 5 shows that  $SP$  accepts 1-packets  $p_{i,1}$  ( $i \in [1, A]$ ), indicated as (5’),  $p_{i,1}$  ( $i \in [B + 1, B + A]$ ), indicated as (6’), and so on. Now, for each  $w \in [1, N]$ ,  $SP$  accepts  $A$  1-packets  $p_{(w-1)B+1,1}, p_{(w-1)B+2,1}, \dots, p_{(w-1)B+A,1}$ , hence  $NA$  1-packets in total, and rejects the rest. Next, for each integer  $j \in [2, k]$  and each integer  $y \in [0, 2^{k-j} - 1]$ ,  $SP$  accepts  $A$   $j$ -packets  $p_{y2^{j-1}D+1,j}, \dots, p_{y2^{j-1}D+A,j}$  but rejects others. Note that the number of  $j$ -packets accepted by  $SP$  is  $2^{k-j}A$ . In particular, the number of  $k$ -packets accepted by  $SP$  is  $A$ . Therefore,  $V_{SP}(\sigma) = A$ .

Next, consider  $OPT$ ’s behavior. Let  $b_1 = 0$  and  $b_z = \sum_{j=1}^{z-1} 2^{k-j-1}D$  for each integer  $z \in [2, k-1]$ .  $OPT$  completes  $f_{b_z+D+1}, \dots, f_{b_z+D+B}$  for each integer  $z \in [1, k-1]$ . Therefore,  $V_{OPT}(\sigma) = (k-1)B$  and  $\frac{V_{OPT}(\sigma)}{V_{SP}(\sigma)} = \frac{(k-1)B}{A} = \Omega(k^2)$  since  $A = \lfloor B/k \rfloor$ .

On the other hand,  $MF$  completes  $f_{b_z+1}, \dots, f_{b_z+\lfloor A/2 \rfloor}$  for each integer  $z \in [1, k-1]$  and  $f_{b_k+\lfloor A/2 \rfloor+1}, \dots, f_{b_k+A}$ . Therefore,  $V_{MF}(\sigma) = (k-1)\lfloor A/2 \rfloor + A - \lfloor A/2 \rfloor \geq k\lfloor A/2 \rfloor$  and  $V_{OPT}(\sigma)/V_{MF}(\sigma) \leq (k-1)B/(k\lfloor A/2 \rfloor) \leq B/\lfloor A/2 \rfloor$  hold.

```

 $t := 0.$ 
for  $w = 1, \dots, N$  do
  1-packets  $p_{(w-1)B+1,1}, \dots, p_{wB,1}$  arrive at the  $t$ th phase.
   $t := t + B.$ 
end for
for  $j = 2, \dots, k$  do
   $t := (j-1)NB.$ 
  for  $y = 0, \dots, 2^{k-j} - 1$  do
     $j$ -packets  $p_{y2^{j-1}D+1,j}, \dots, p_{y2^{j-1}D+2^{j-2}D+D,j}$  arrive at the  $t$ th phase.
     $t := t + B.$ 
    for  $x = 1, \dots, 2^{j-2} - 1$  do
       $j$ -packets  $p_{y2^{j-1}D+2^{j-2}D+xD+1,j}, \dots, p_{y2^{j-1}D+2^{j-2}D+xD+D,j}$  arrive at the  $t$ th phase.
       $t := t + B.$ 
    end for
  end for
end for

```

Figure 4: Pseudocode of arriving packets in  $\sigma$

## B Execution Example of $MF$

In this section, we give an execution example of  $MF$  for a given input  $\sigma$  in Tables 1 and 2. We suppose that  $k = 3$  and  $B = 12$ , which means  $A = 12/3 = 4$ .  $\sigma$  includes 120 frames  $f_1, \dots, f_{120}$ . For each  $i (\in [1, 120])$ ,  $p_i, q_i$  and  $r_i$  denote the 1-packet, 2-packet and 3-packet in  $f_i$ , respectively. We suppose that  $\text{arr}(p_1) \leq \text{arr}(p_2) \leq \dots \leq \text{arr}(p_{120})$ . All the 1-packets (all the 2-packets and 3-packets) arrive as shown in Table 1 (Table 2). Columns starting from the left in the tables present the arrival times of packets, the names of arriving packets, actions by  $GR_1$  for arriving packets (only in Table 1), actions by  $MF$  for arriving packets, the names of cases executed by  $MF$  and the block numbers of arriving packets (only in Table 1).

For example, 1-packets  $p_1, p_2, p_3$  and  $p_4$  arrive at phase 0,  $MF$  executes Case 1.2.1, and accepts these packets. (See Figure 10.) The block numbers of these 1-packets are 1. In particular,  $MF$  accepts 2-packet  $q_{85}$  at phase 120, and preempts  $q_{51}$  that is stored in its buffer at  $t_{q_{85}} -$ . (That is,  $MF$  discards  $q_{51}$  using a “middle-drop” policy.) Moreover, when  $MF$  accepts 3-packet  $r_{85}$  at the 120th phase,  $MF$  executes Case 2.2.2, and preempts  $r_{49}$ . Hence,  $f_{49}$  becomes invalid for  $MF$ . At this time,  $MF$  preempts 2-packet  $q_{49}$  as well. In addition, the frames  $f$  such that the 1-packets in  $f$  are accepted by  $MF$ , and  $g(f) = 2$  are  $f_{49}, f_{50}, f_{51}$  and  $f_{52}$ . At event times  $t_{q_{85}}$  and  $t_{q_{86}}$ ,  $q_{51}$  and  $q_{52}$  are preempted, respectively. That is, the number of valid frames of  $MF$  with block number 2 decreases to less than  $\lfloor A/2 \rfloor = 2$  at event time  $t_{r_{85}}$ . Thus,  $MF$  further executes Case 2.2.2.1, and preempts all the packets whose block numbers are 2 in its buffer.

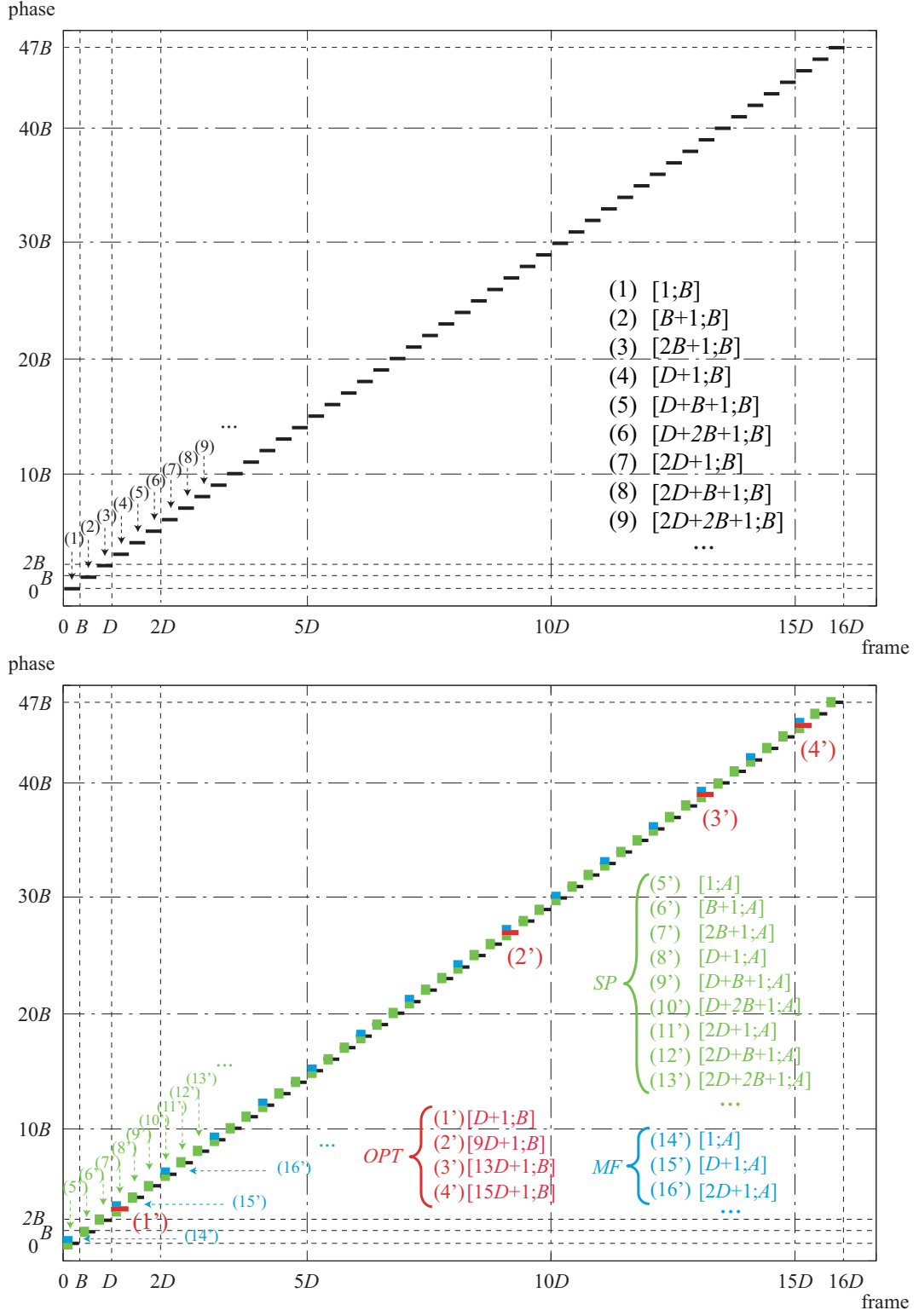


Figure 5: Arriving 1-packets in  $\sigma$  when  $k = 5$ .  $[i; x]$  in the figure denotes all the 1-packets in  $f_i, \dots, f_{i+x-1}$  for any integers  $i$  and  $x$ .

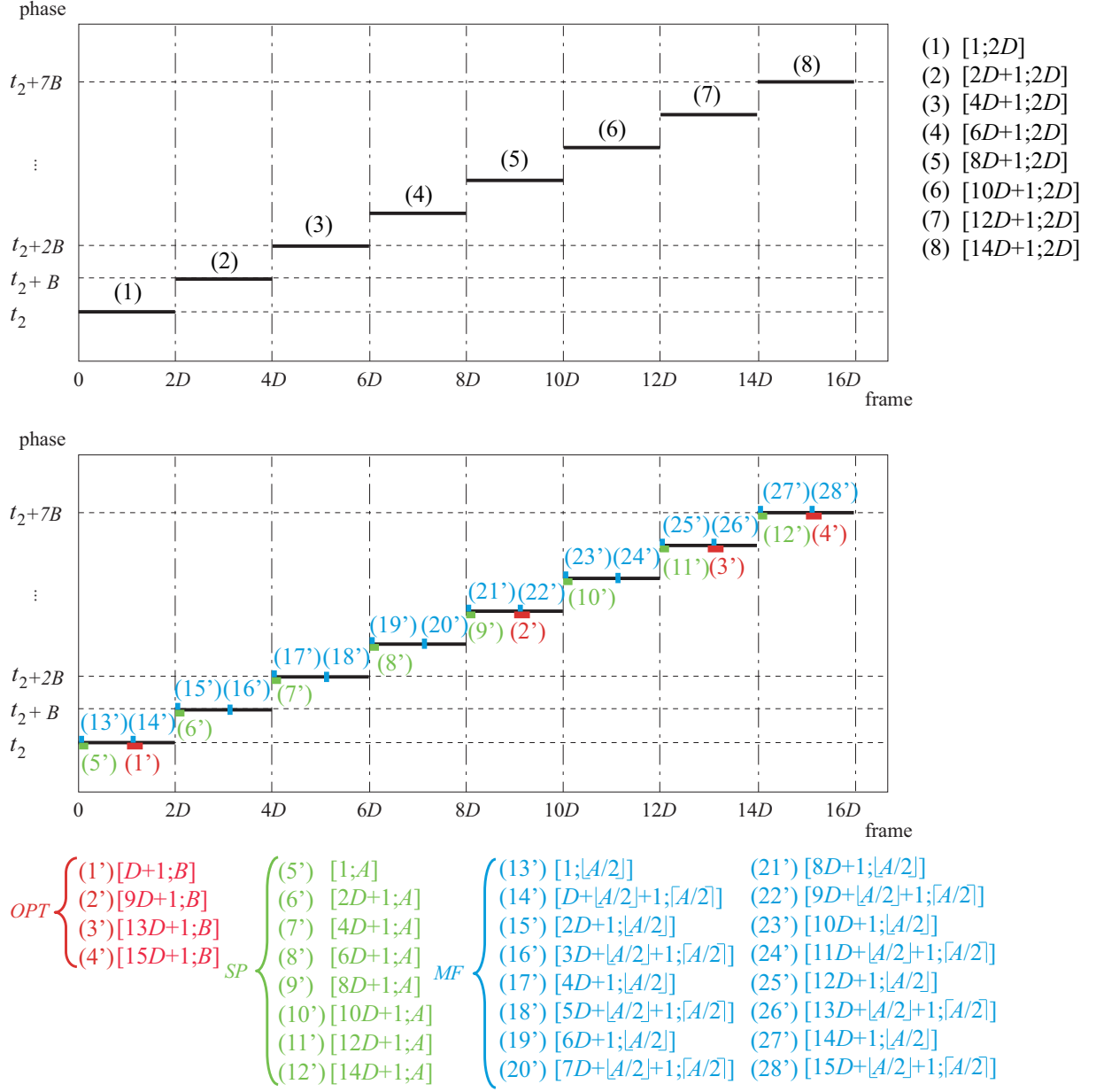


Figure 6: Arriving 2-packets in  $\sigma$  when  $k = 5$

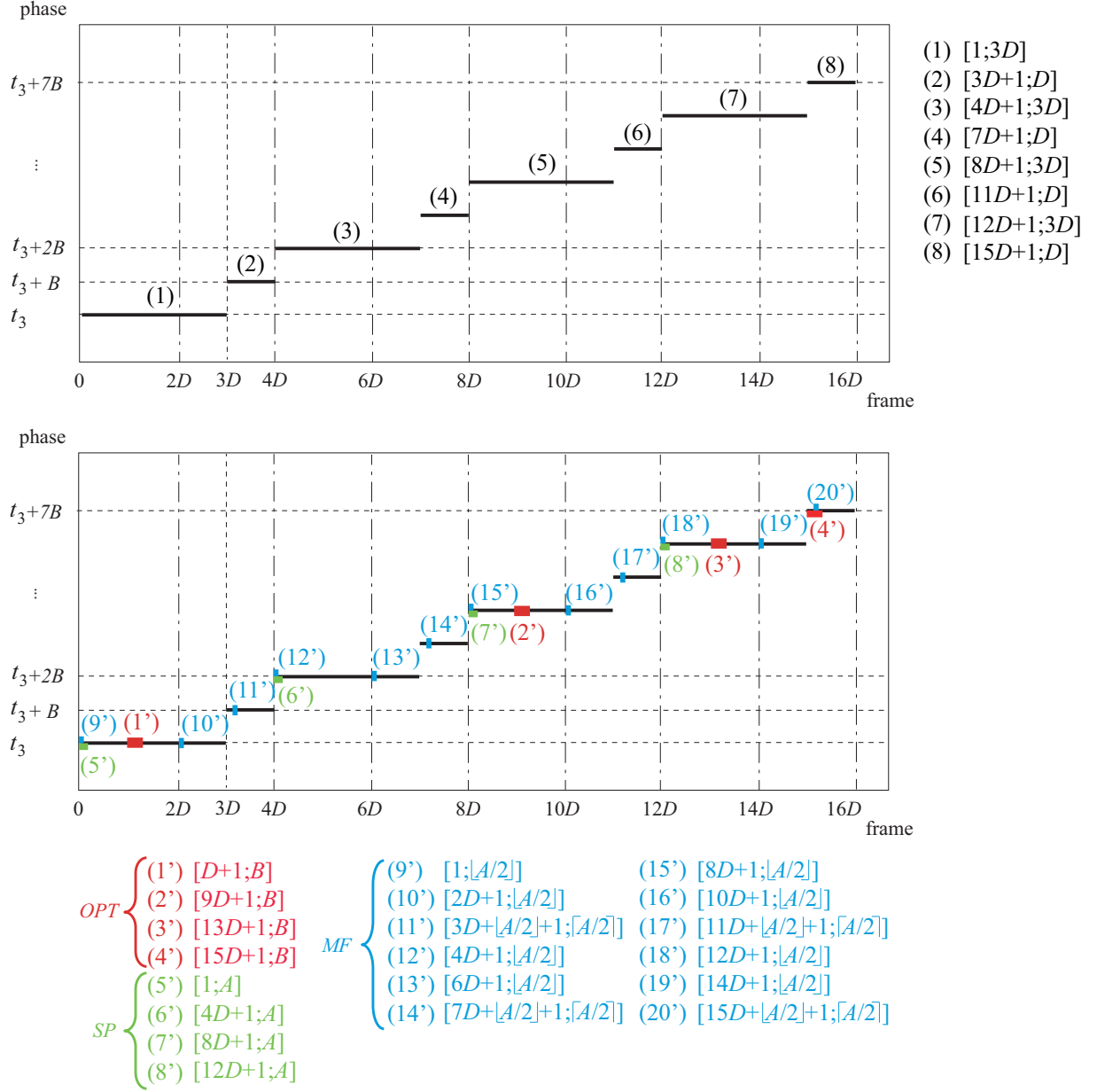


Figure 7: Arriving 3-packets in  $\sigma$  when  $k = 5$

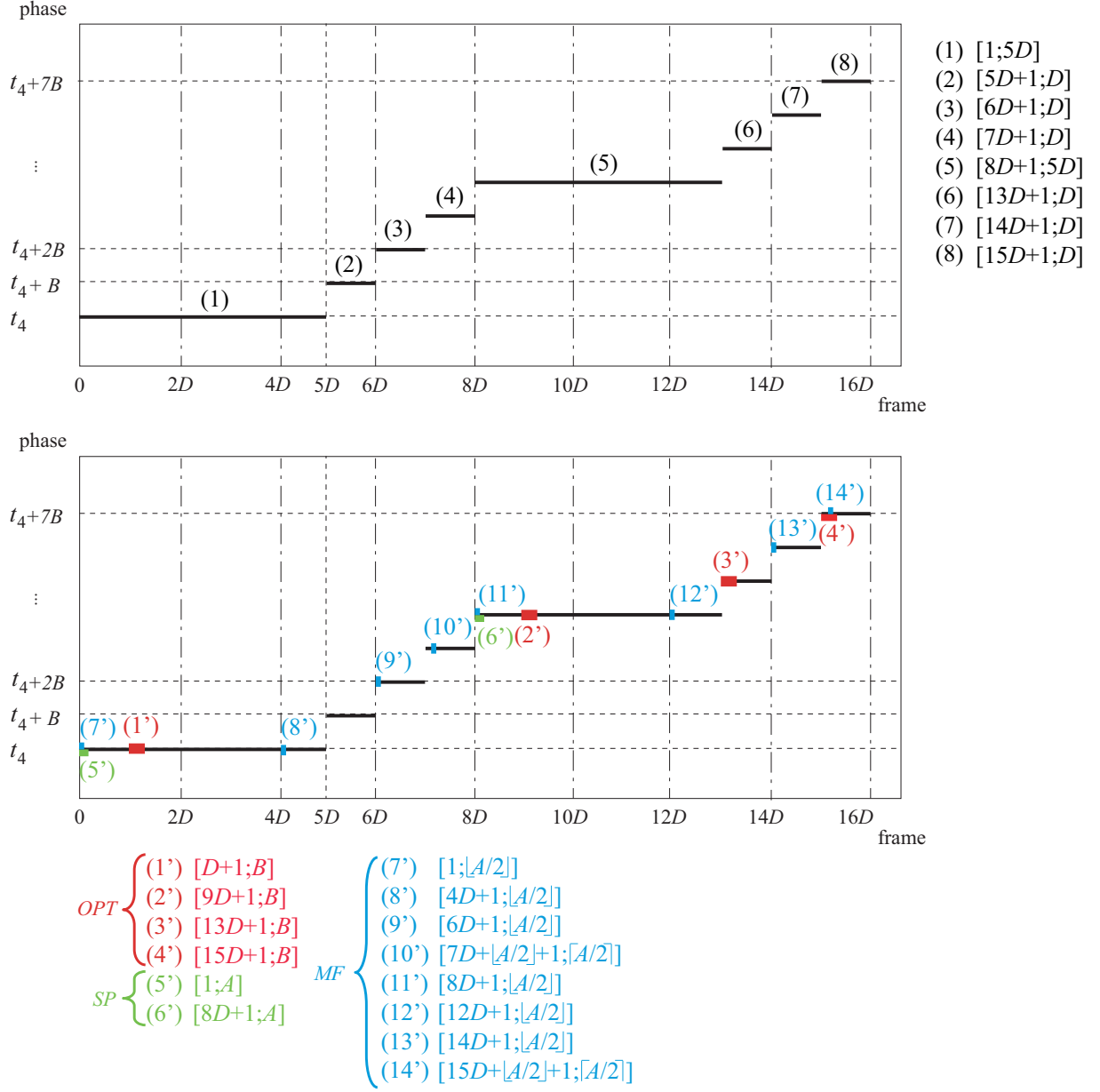


Figure 8: Arriving 4-packets in  $\sigma$  when  $k = 5$

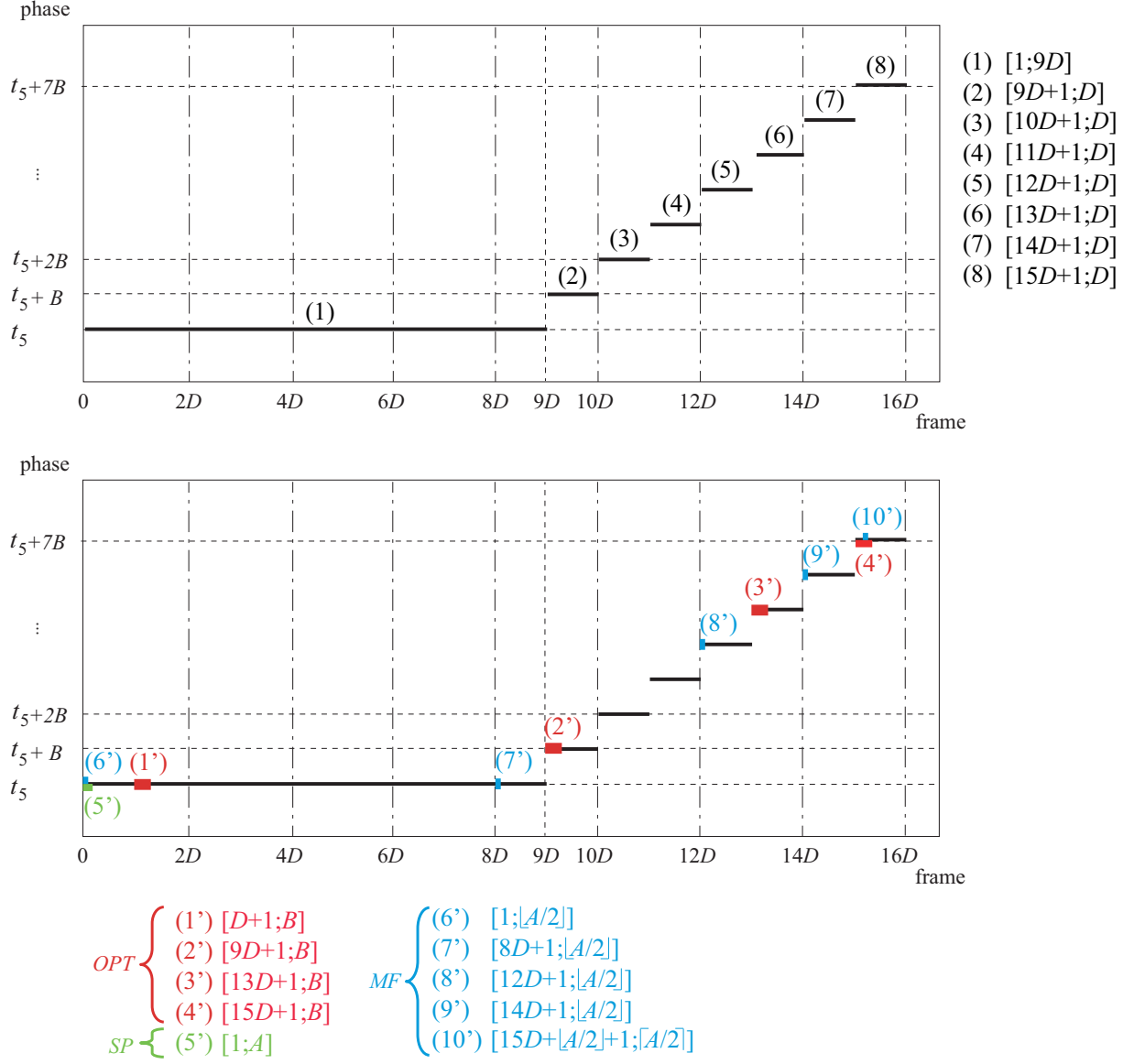


Figure 9: Arriving 5-packets in  $\sigma$  when  $k = 5$

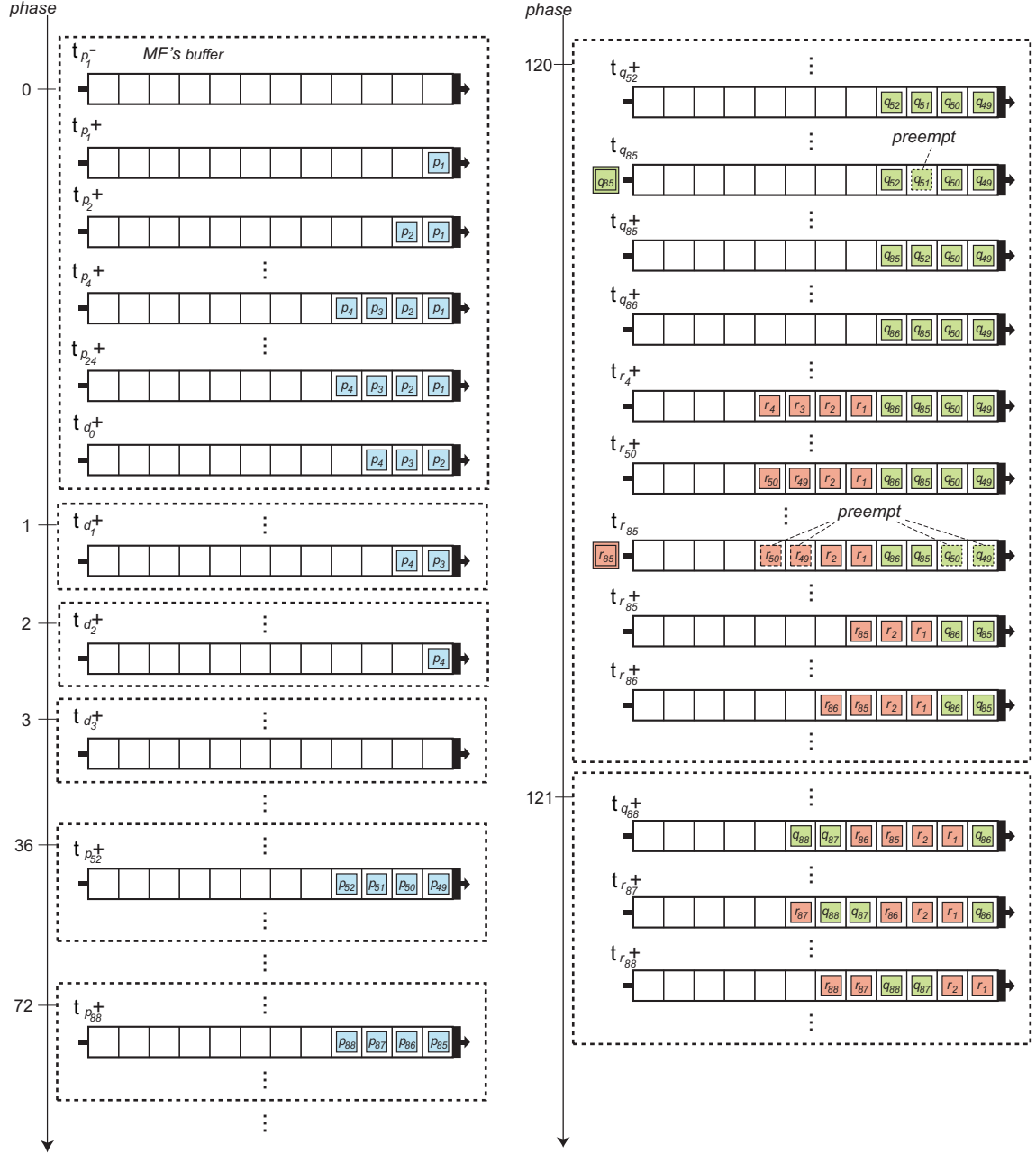


Figure 10: Execution example of  $MF$ .  $t_{d_i}$  denotes the  $i$ th delivery time.

Table 1: Arriving 1-packets in  $\sigma$

| Time | Arrival<br>Packets        | $GR_1$ 's<br>Action | $MF$ 's<br>Action | Case  | Block |
|------|---------------------------|---------------------|-------------------|-------|-------|
| 0    | $p_1, \dots, p_4$         | accept              | accept            | 1.2.1 | 1     |
|      | $p_5, \dots, p_{12}$      | accept              | reject            | 1.2.2 | 1     |
|      | $p_{13}, \dots, p_{24}$   | reject              | reject            | 1.1   | 1     |
| 12   | $p_{25}, \dots, p_{36}$   | accept              | reject            | 1.2.2 | 1     |
| 24   | $p_{37}, \dots, p_{47}$   | accept              | reject            | 1.2.2 | 1     |
|      | $p_{48}$                  | accept              | reject            | 1.2.3 | 1     |
| 36   | $p_{49}, \dots, p_{52}$   | accept              | accept            | 1.2.1 | 2     |
|      | $p_{53}, \dots, p_{60}$   | accept              | reject            | 1.2.2 | 2     |
| 48   | $p_{61}, \dots, p_{72}$   | accept              | reject            | 1.2.2 | 2     |
| 60   | $p_{73}, \dots, p_{83}$   | accept              | reject            | 1.2.2 | 2     |
|      | $p_{84}$                  | accept              | reject            | 1.2.3 | 2     |
| 72   | $p_{85}, \dots, p_{88}$   | accept              | accept            | 1.2.1 | 3     |
|      | $p_{89}, \dots, p_{96}$   | accept              | reject            | 1.2.2 | 3     |
| 84   | $p_{97}, \dots, p_{108}$  | accept              | reject            | 1.2.2 | 3     |
| 96   | $p_{109}, \dots, p_{119}$ | accept              | reject            | 1.2.2 | 3     |
|      | $p_{120}$                 | accept              | reject            | 1.2.3 | 3     |

Table 2: Arriving 2-packets and 3-packets in  $\sigma$

| Time | Arrival<br>Packets       | $MF$ 's<br>Action                                                       | Case             |
|------|--------------------------|-------------------------------------------------------------------------|------------------|
| 108  | $q_1, \dots, q_4$        | accept                                                                  | 2.2.1            |
|      | $q_5, \dots, q_{48}$     | reject                                                                  | 2.1              |
| 120  | $q_{49}, \dots, q_{52}$  | accept                                                                  | 2.2.1            |
|      | $q_{53}, \dots, q_{84}$  | reject                                                                  | 2.1              |
|      | $q_{85}$                 | preempt $q_{51}$<br>accept $q_{85}$                                     | 2.2.2            |
|      | $q_{86}$                 | preempt $q_{52}$<br>accept $q_{86}$                                     | 2.2.2            |
|      | $r_1, \dots, r_4$        | accept                                                                  | 2.2.1            |
|      | $r_5, \dots, r_{48}$     | reject                                                                  | 2.1              |
|      | $r_{49}$                 | preempt $r_3$<br>accept $r_{49}$                                        | 2.2.2            |
|      | $r_{50}$                 | preempt $r_4$<br>accept $r_{50}$                                        | 2.2.2            |
|      | $r_{51}, \dots, r_{84}$  | reject                                                                  | 2.1              |
|      | $r_{85}$                 | preempt $r_{49}, q_{49}$<br>accept $r_{85}$<br>preempt $r_{50}, q_{50}$ | 2.2.2<br>2.2.2.1 |
|      | $r_{86}$                 | accept                                                                  | 2.2.1            |
| 121  | $q_{87}, q_{88}$         | accept                                                                  | 2.2.1            |
|      | $q_{89}, \dots, q_{120}$ | reject                                                                  | 2.1              |
|      | $r_{87}$                 | preempt $r_{85}$<br>accept $r_{87}$                                     | 2.2.2            |
|      | $r_{88}$                 | preempt $q_{86}, r_{86}$<br>accept $r_{88}$                             | 2.2.2            |
|      | $r_{89}, \dots, r_{120}$ | reject                                                                  | 2.1              |