Computer Networks xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Tight analysis of priority queuing for egress traffic☆

Jun Kawahara [a,*], Koji M. Kobayashi [b], Tomotaka Maeda [c]

[a] Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara 6300192, Japan
[b] National Institute of Informatics, Japan
[c] Academic Center for Computing and Media Studies, Kyoto University, Japan

ARTICLE INFO

Article history:
Received 12 January 2015
Revised 2 September 2015
Accepted 3 September 2015
Available online xxx

Keywords:
Online problem
Competitive analysis
Quality of service
Priority queuing
Egress traffic control
Multi-queue switch

ABSTRACT

Recently, the problems of evaluating performances of switches and routers have been formulated as online problems, and a great amount of results have been presented. In this paper, we focus on managing outgoing packets (called *egress traffic*) on switches that support Quality of Service (QoS), and analyze the performance of one of the most fundamental scheduling policies *Priority Queuing* (*PQ*) using competitive analysis. We formulate the problem of managing egress queues as follows: An output interface is equipped with m queues, each of which has a buffer of size B. The size of a packet is unit, and each buffer can store up to B packets simultaneously. Each packet is associated with one of m priority values α_j $(1 \le j \le m)$, where $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_m$, $\alpha_1 = 1$, and $\alpha_m = \alpha$ and the task of an online algorithm is to select one of m queues at each scheduling step. The purpose of this problem is to maximize the sum of the values of the scheduled packets.

For any B and any m, we show that the competitive ratio of *PQ* is exactly $2 - \min_{x \in [1, m-1]} \{ \frac{\alpha_{x+1}}{\sum_{j=1}^{x+1} \alpha_j} \}$. That is, we conduct a complete analysis of the performance of *PQ* using worst case analysis. Moreover, we show that no deterministic online algorithm can have a competitive ratio smaller than $1 + \frac{\alpha^3+\alpha^2+\alpha}{\alpha^4+4\alpha^3+3\alpha^2+4\alpha+1}$.

1. Introduction

In recent years, the Internet has provided a rich variety of applications, such as teleconferencing, video streaming, IP telephone, mainly thanks to the rapid growth of the broadband technology. To enjoy such services, the demand for the Quality of Service (QoS) guarantee is crucial. For example, usually there is little requirement for downloading programs or picture images, whereas real-time services, such as distance meeting, require constant-rate packet transmission. One possible way of supporting QoS is differentiated services (Diffserv) [15]. In DiffServ, a value is assigned to each packet according to the importance of the packet. Then, switches that support QoS (QoS switches) decide the order of packets to be processed, based on the value of packets. In such a mechanism, one of the main issues in designing algorithms is how to treat packets depending on the priority in buffering or scheduling. This kind of problems was recently modeled as an *online problem*, and the *competitive analysis* [16,40] of algorithms has been done.

Aiello et al. [1] was the first to attempt this study, in which they considered a model with only one First In First Out (FIFO) queue. This model mainly focuses on the buffer management issue of the input port of QoS switches: There is one FIFO queue of size B, meaning that it can store up to B packets. An input is a sequence of events. An event is either an *arrival event*, at which a packet with a specified priority value arrives, or a *scheduling event*, at which the packet at the head of the queue will be transmitted. The task of an online

☆ A preliminary version of this paper was presented at the 8th annual international conference on combinatorial optimization and applications, COCOA 2014.

* Corresponding author. Tel.: +81 -743 -72 -5362; fax: +81743725369.
E-mail address: jkawahara@is.naist.jp (J. Kawahara).

http://dx.doi.org/10.1016/j.comnet.2015.09.001
1389-1286/© 2015 Published by Elsevier B.V.

(buffer management) algorithm is to decide, when a packet arrives at an arrival event, whether to accept or to reject it (in order to keep a room for future packets with higher priority). The purpose of the problem is to maximize the sum of the values of the transmitted packets. Aiello et al. analyzed the competitiveness of the Greedy Policy, the Round Robin Policy, the Fixed Partition Policy, etc.

After the publication of this seminal paper, more and more complicated models have been introduced and studied, some of which are as follows: Azar et al. [9] considered the *multi-queue switch model*, which formulates the buffering problem of one input port of the switch. In this problem, an input port has N input buffers connected to a common output buffer. The task of an online algorithm is now not only buffer management but also scheduling. At each scheduling event, an algorithm selects one of N input buffers, and the packet at the head of the selected buffer is transmitted to the inside of the switch through the output buffer. There are some formulations that model not only one port but the entire switch. For example, Kesselman et al. [29] introduced the *Combined Input and Output Queue (CIOQ) switch model*. In this model, a switch consists of N input ports and N output ports, where each port has a buffer. At an *arrival phase*, a packet (with the specified destination output port) arrives at an input port. The task of an online algorithm is buffer management as mentioned before. At a *transmission phase*, all the packets at the top of the nonempty buffers of output ports are transmitted. Hence, there is no task of an online algorithm. At a *scheduling phase*, packets at the top of the buffers of input ports are transmitted to the buffers of the output ports. Here, an online algorithm computes a matching between input ports and output ports. According to this matching, the packets in the input ports will be transmitted to the corresponding output ports. Kesselman et al. [32] considered the *crossbar switch model*, which models the scheduling phase of the CIOQ switch model more in detail. In this model, there is also a buffer for each pair of an input port and an output port. Thus, there arises another buffer management problem at scheduling phases.

In some real implementation (e.g., [17]), additional buffers are equipped with each output port of a QoS switch to control the outgoing packets (called *egress traffic*). Assume that there are m priority values of packets $\alpha_1, \alpha_2, \dots, \alpha_m$ such that $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_m$. Then, m FIFO queues $Q^{(1)}, Q^{(2)}, \dots, Q^{(m)}$ are introduced for each output port, and a packet with the value α_i arriving at this output port is stored in the queue $Q^{(i)}$. Usually, this buffering policy is greedy, namely, when a packet arrives, it is rejected if the corresponding queue is full, and accepted otherwise. The task of an algorithm is to decide which queue to transmit a packet at each scheduling event.

Several practical algorithms, such as Priority Queuing (*PQ*), Weighted Round-Robin (*WRR*) [25], and Weighted Fair Queuing (*WFQ*) [20], are currently implemented in network switches. *PQ* is the most fundamental algorithm, which selects the highest priority non-empty queue. This policy is implemented in many switches by default. (e.g., Cisco's Catalyst 2955 series [18]) In the *WRR* algorithm, queues are selected according to the round robin policy based on the weight of packets corresponding to queues, i.e., the rate of selecting $Q^{(i)}$ in one round is proportional to α_i for each i. This algorithm is implemented in Cisco's Catalyst 2955 series [18] and so on. In the *WFQ* algorithm, length of packets, as well as the priority values, are taken into consideration so that shorter packets are more likely to be scheduled. This algorithm is implemented in Cisco's Catalyst 6500 series [19] and so on.

In spite of intensive studies on online buffer management and scheduling algorithms, to the best of our knowledge, there have been no research on the egress traffic control, which we focus on in this paper. Our purpose is to evaluate the performances of actual scheduling algorithms for egress queues.

Our Results. We formulate this problem as an online problem, and provide a tight analysis of the performance of *PQ* using competitive analysis. Specifically, for any B, we show that the competitive ratio of *PQ* is exactly $2 - \min_{x \in [1, m-1]}\{\frac{\alpha_{x+1}}{\sum_{j=1}^{x+1} \alpha_j}\}$. *PQ* is trivial to implement, and has a lower computational load than the other policies, such as *WRR* and *WFQ*. Hence, it is meaningful to analyze the exact performance of *PQ*. Moreover, we present a lower bound of $1 + \frac{\alpha^3 + \alpha^2 + \alpha}{\alpha^4 + 4\alpha^3 + 3\alpha^2 + 4\alpha + 1}$ on the competitive ratio of any deterministic algorithm.

Related Work. Independently of our work, Al-Bawani and Souza [2] have very recently considered much the same model. *PQ* is called the greedy algorithm in their paper. Unlike our setting, they discussed only the case where any two of the values differ, that is, $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_m$. Also, they assumed that for any $j (\in [1, m])$, the jth queue can store at most $B_j (\in [1, B])$ packets at a time. In the case of $B_j = B$, that is, in the same setting as ours, they showed that the competitive ratio of *PQ* is at most $2 - \min_{j \in [1, m-1]}\{\frac{\alpha_{j+1} - \alpha_j}{\alpha_{j+1}}\}$ for any m and B. When comparing our result and their upper bound, we have $2 - \min_{x \in [1, m-1]}\{\frac{\alpha_{x+1}}{\sum_{j=1}^{x+1} \alpha_j}\} < 2 - \min_{j \in [1, m-1]}\{\frac{\alpha_{j+1} - \alpha_j}{\alpha_{j+1}}\}$ by elementary calculation (see Appendix A in Appendix). Note that $2 - \min_{j \in [1, m-1]}\{\frac{\alpha_{j+1} - \alpha_j}{\alpha_{j+1}}\}$ is equal to 2 when there exists some z such that $\alpha_{z+1} = \alpha_z$. In general practical switches, the sizes of any two egress queues attached to the same output port are equivalent by default. Since we focus on evaluating the performance of algorithms in a more practical setting (which might be less generalized), we assume that the size of each queue is B. Moreover, our analysis in this paper does not depend on the maximum numbers of packets stored in buffers, and instead it depends on whether buffers are full of packets. Thus, the exact competitive ratio of *PQ* would be derived for the setting where for any j, the size of the jth queue is B_j in the same way as this paper. (If we apply our method in their setting, Lemma 3.7 in Section 3.3 has to be fixed slightly. However the competitive ratio obtained in this setting seems to be a more complicated value including some min s or max es.)

As mentioned earlier, there are a lot of studies concentrating on evaluating performances of functions of switches and routers, such as queue management and packet scheduling. The most basic one is the model consisting of single FIFO queue by Aiello et al. [1] mentioned above. In their model, each packet can take one of two values 1 or $\alpha (> 1)$. Andelman et al. [7] generalized the values of packets to any value between 1 and α. Another generalization is to allow *preemption*, namely, one may drop a packet that is already stored in a queue. Results of the competitiveness on this

Please cite this article as: J. Kawahara et al., Tight analysis of priority queuing for egress traffic, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.09.001

model are given in [1,5–7,21,26,28,41]. Recently Kogan et al. [38] analyzed the performance of some packet scheduling policies for single FIFO queue built on processing cycles and conducted some simulation research for the policies.

The multi-queue switch model [9,11,36] consists of m FIFO queues. In this model, the task of an algorithm is to manage its buffers and to schedule packets. The problem of designing only a scheduling algorithm in multi-queue switches is considered in [4,8,13,14,35]. Moreover, Albers and Jacobs [3] performed an experimental study for the first time on several online scheduling algorithms for this model. Also, the overall performance of several switches, such as shared-memory switches [24,27,34], CIOQ switches [10,29,30,33], and crossbar switches [31,32], are extensively studied.

Fleischer and Koga [22] and Bar-Noy et al. [12] studied the online problem of minimizing the length of the longest queue in a switch, in which the size of each queue is unbounded. In [22] and [12], they showed that the competitive ratio of any online algorithm is $\Omega(\log m)$, where m is the number of queues in a switch. Fleischer and Koga [22] presented a lower bound of $\Omega(m)$ for the round robin policy. In addition, in [22] and [12], the competitive ratio of a greedy algorithm called Longest Queue First is $O(\log m)$. Recently, Kogan et al. [37] studied a multi-queue switch where packets with different required processing times arrive. (In the other settings mentioned above, the required processing times of all packets are equivalent.)

Furthermore, some comprehensive surveys showed much research on buffer management and scheduling policies (see e.g. [23,39]).

2. Model description

In this section, we formally define the problem studied in this paper. Our model consists of m queues, each with a buffer of size B. The size of a packet is unit, which means that each buffer can store up to B packets simultaneously. Each packet is associated with one of m values α_i $(1 \le i \le m)$, which represents the priority of this packet where a packet with larger value is of higher priority. Without loss of generality, we assume that $\alpha_1 = 1$, $\alpha_m = \alpha$, and $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_m$. The ith queue is denoted $Q^{(i)}$ and is also associated with its priority value α_i. An arriving packet with the value α_i is stored in $Q^{(i)}$.

An input for this model is a sequence of *events*. Each event is an *arrival event* or a *scheduling event*. At an arrival event, a packet arrives at one of m queues, and the packet is *accepted* to the buffer when the corresponding queue has free space. Otherwise, it is *rejected*. If a packet is accepted, it is stored at the tail of the corresponding queue. At a scheduling event, an online algorithm selects one non-empty queue and transmits the packet at the head of the selected queue. We assume that any input contains enough scheduling events to transmit all the arriving packets in it. That is, any algorithm can certainly transmit a packet stored in its queue. Note that this assumption is common in the buffer management problem. (See e.g. [23].) The *gain* of an algorithm is the sum of the values of transmitted packets. Our goal is to maximize it. The gain of an algorithm *ALG* for an input σ is denoted by $V_{ALG}(\sigma)$. If $V_{ALG}(\sigma) \ge V_{OPT}(\sigma)/c$ for an arbitrary input σ, we say that *ALG* is *c-competitive*, where *OPT* is an optimal offline algorithm for σ.

3. Analysis of priority queuing

3.1. Priority queuing

PQ is a greedy algorithm. At a scheduling event, *PQ* selects the non-empty queue with the largest index. For analysis, we assume that *OPT* does not reject an arriving packet. This assumption does not affect the analysis of the competitive ratio. (See Lemma B.1 in Appendix B.)

3.2. Overview of the analysis

We define an *extra packet* as a packet which is accepted by *OPT* but rejected by *PQ*. In the following analysis, we evaluate the sum of the values of extra packets to obtain the competitive ratio of *PQ*. We introduce some notation for our analysis. For any input σ, $k_j(\sigma)$ denotes the number of extra packets arriving at $Q^{(j)}$ when treating σ. We call a queue at which at least one extra packet arrives a *good queue* when treating σ. $n(\sigma)$ denotes the number of good queues for σ. Moreover, for any input σ and any $i(\in [1, n(\sigma)])$, $q_i(\sigma)$ denotes the good queue with the ith minimum index. That is, $1 \le q_1(\sigma) < q_2(\sigma) < \cdots < q_{n(\sigma)}(\sigma) \le m$. Also, we define $q_{n(\sigma)+1}(\sigma) = m$. In addition, for any input σ, $s_j(\sigma)$ denotes the number of packets which *PQ* transmits from $Q^{(j)}$. We drop the input σ from the notation when it is clear. Then, $V_{PQ}(\sigma) = \sum_{j=1}^{m} \alpha_j s_j$, and $V_{OPT}(\sigma) = V_{PQ}(\sigma) + \sum_{i=1}^{n} \alpha_{q_i} k_{q_i}$. (The equality follows from the assumption that *OPT* does not reject any packet, which is proven in Lemma B.1.)

First, we show that $k_m = 0$, that is, $q_n + 1 \le m$, in Lemma 3.2. We will gradually construct some input set $\mathcal{S}^*$ (defined below) from Lemma 3.4 –Lemma 3.9 using some adversarial strategies against *PQ*. Moreover, in Lemma 3.10, we prove that the set $\mathcal{S}^*$ includes an input σ such that the ratio $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)}$ is maximized. That is, we show that there exists an input σ^* in the set $\mathcal{S}^*$ to get the competitive ratio of *PQ* in the lemma. More formally, we define the set $\mathcal{S}^*$ of the inputs σ' satisfying the following five conditions: (i) for any $i(\in [1, n(\sigma') - 1])$, $q_i(\sigma') + 1 = q_{i+1}(\sigma')$, (ii) for any $i(\in [1, n(\sigma')])$, $k_{q_i(\sigma')}(\sigma') = B$, (iii) for any $j(\in [q_1(\sigma'), q_{n(\sigma')}(\sigma') + 1])$, $s_j(\sigma') = B$, (iv) for any $j(\in [1, q_1(\sigma') - 1])$, $s_j(\sigma') = 0$ if $q_1(\sigma') - 1 \ge 1$, and (v) for any $j(\in [q_{n(\sigma')}(\sigma') + 2, m])$, $s_j(\sigma') = 0$ if $q_{n(\sigma')}(\sigma') + 2 \le m$. Then, we show that there exists an input $\sigma^* \in \mathcal{S}^*$ such that $\max_{\sigma''}\{\frac{V_{OPT}(\sigma'')}{V_{PQ}(\sigma'')}\} = \frac{V_{OPT}(\sigma^*)}{V_{PQ}(\sigma^*)}$ in Lemma 3.10.

By the above lemmas, we can obtain the competitive ratio of *PQ* as follows: For ease of presentation, we write $s_i(\sigma^*)$, $n(\sigma^*)$, $q_i(\sigma^*)$ and $k_i(\sigma^*)$ as s_i^*, n^*, q_i^* and k_i^*, respectively. Thus, $\frac{V_{OPT}(\sigma^*)}{V_{PQ}(\sigma^*)} = \frac{V_{PQ}(\sigma^*) + \sum_{i=1}^{n^*} \alpha_{q_i^*} k_{q_i^*}^*}{V_{PQ}(\sigma^*)} = 1 + \frac{B \sum_{j=q_1^*}^{q_{n^*}^*} \alpha_j}{B \sum_{j=q_1^*}^{q_{n^*}^*+1} \alpha_j} \le 1 + \frac{\sum_{j=1}^{q_{n^*}^*} \alpha_j}{\sum_{j=1}^{q_{n^*}^*+1} \alpha_j} = 2 - \frac{\alpha_{q_{n^*+1}}}{\sum_{j=1}^{q_{n^*}+1} \alpha_j}$. The last inequality follows from $\frac{\sum_{j=x-1}^{y} \alpha_j}{\sum_{j=x-1}^{y+1} \alpha_j} - \frac{\sum_{j=x}^{y} \alpha_j}{\sum_{j=x}^{y+1} \alpha_j} = (\sum_{j=x-1}^{y} \alpha_j \sum_{j=x}^{y+1} \alpha_j -$

Please cite this article as: J. Kawahara et al., Tight analysis of priority queuing for egress traffic, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.09.001

$\sum_{j=x}^{y}\alpha_j\sum_{j=x-1}^{y+1}\alpha_j)/(\sum_{j=x-1}^{y+1}\alpha_j\sum_{j=x}^{y+1}\alpha_j) = (\alpha_{x-1}\alpha_{y+1})/(\sum_{j=x-1}^{y+1}\alpha_j\sum_{j=x}^{y+1}\alpha_j) > 0$. This gives an upper bound on the competitive ratio of *PQ*.

On the other hand, we show that there exists some input $\hat{\sigma}$ such that $\frac{V_{OPT}(\hat{\sigma})}{V_{PQ}(\hat{\sigma})} = 2 - \min_{x\in[1,m-1]}\{\frac{\alpha_{x+1}}{\sum_{j=1}^{x+1}\alpha_j}\}$ in Lemma 3.11, which presents a lower bound for *PQ*. Therefore, we have the following theorem:

Theorem 3.1. *The competitive ratio of PQ is exactly* $2 - \min_{x\in[1,m-1]}\{\frac{\alpha_{x+1}}{\sum_{j=1}^{x+1}\alpha_j}\}$.

3.3. Competitive analysis of PQ

We give some definitions. For ease of presentation, an *event time* denotes a time when an event happens, and any other moment is called a *non-event time*. We assign index numbers 1 through *B* to each position of a queue from the head to the tail in increasing order. The *j*th position of $Q^{(i)}$ is called the *j*th *cell*. For any non-event time *t*, suppose that the *j*th cell in $Q^{(i)}$ of *PQ* holds a packet at *t* but the *j*th cell *c* in $Q^{(i)}$ of *OPT* does not at *t*. Then, we call *c* a *free* cell at *t*. Note that any extra packet is accepted at a free cell. For any non-event time *t*, let $h_{ALG}^{(j)}(t)$ denote the number of packets which an algorithm *ALG* stores in $Q^{(j)}$ at *t*. We first prove the following lemma. (The lemma is similar to Lemma 2.3 in [2].)

Lemma 3.2. $k_m = 0$.

Proof. By the definition of *PQ*, *PQ* selects the non-empty queue with the highest priority. Thus, $h_{PQ}^{(m)}(t) \le h_{OPT}^{(m)}(t)$ holds at any non-event time *t*. Therefore, there is no free cell in $Q^{(m)}$ of *OPT* at any time. Since any extra packet is accepted to a free cell, $k_m = 0$. □

Next, in order to evaluate the total number of extra packets accepted at each $Q^{(q_i)}$ $(i \in [1, n])$, we construct some matching between extra packets and *PQ*'s packets according to the matching routine defined later. (Note that evaluating the number of extra packets is related to the property (ii) of $\mathcal{S}^*$.) Suppose that extra packet *p* is matched with *PQ*'s packet *p*′ such that *p* and *p*′ are transmitted from $Q^{(i)}$ and $Q^{(i')}$, respectively. Then, the routine constructs this matching where $i < i'$. Let us explain how to construct the matching. We match extra packet one by one with time. However, it is difficult to match an extra packet with *PQ*'s packet in a direct way. Thus, the matching is formed in two stages. That is, at first, for any free cell *c*, we match *c* with some *PQ*'s packet *p* when *c* becomes free at an event time. At a later time, we rematch the extra packet *p*′ accepted into *c* with *p* at an event time when *OPT* accepts *p*′.

In order to realize such matching, we first verify a change in the number of free cells at each event before introducing our matching routine. We give some definitions for that reason. For any event time *t*, *t*− denotes the non-event time before *t* and after the previous event time. Also, *t*+ denotes the non-event time after *t* and before the next event time. The reason why we introduce such notation is that we avoid unclear proofs and that we rigorously specify the location of each packet in a buffer shortly before or after a moment when an algorithm processes (i.e., accepts or rejects) or transmits a packet. Let $f^{(j)}(t)$ denote the number of free cells in $Q^{(j)}$ at a non-event time *t*, that is, $f^{(j)}(t) = \max\{h_{PQ}^{(j)}(t) - h_{OPT}^{(j)}(t), 0\}$. Note that *OPT* does not reject any packet by our assumption (Lemma B.1 in Appendix B). Thus, for any non-event time *t*, $\sum_{j=1}^{m} h_{OPT}^{(j)}(t) > 0$ if $\sum_{j=1}^{m} h_{PQ}^{(j)}(t) > 0$.

Arrival event: Let *p* be the packet arriving at $Q^{(x)}$ at an event time *t*.

Case A1: Both *PQ* and *OPT* accept *p*, and $h_{PQ}^{(x)}(t-) - h_{OPT}^{(x)}(t-) > 0$: Since $h_{PQ}^{(x)}(t+) = h_{PQ}^{(x)}(t-) + 1$ and $h_{OPT}^{(x)}(t+) = h_{OPT}^{(x)}(t-) + 1$, $h_{PQ}^{(x)}(t+) - h_{OPT}^{(x)}(t+) > 0$. Thus, the $(h_{PQ}^{(x)}(t-) + 1)$st cell of $Q^{(x)}$ becomes free in place of the $(h_{OPT}^{(x)}(t-) + 1)$st cell of $Q^{(x)}$. Hence $f^{(x)}(t+) = f^{(x)}(t-)$.

Case A2: Both *PQ* and *OPT* accept *p*, and $h_{PQ}^{(x)}(t-) - h_{OPT}^{(x)}(t-) \le 0$: Since $h_{PQ}^{(x)}(t+) = h_{PQ}^{(x)}(t-) + 1$ and $h_{OPT}^{(x)}(t+) = h_{OPT}^{(x)}(t-) + 1$, $h_{PQ}^{(x)}(t+) - h_{OPT}^{(x)}(t+) \le 0$. Since the states of all the free cells do not change before and after *t*, $f^{(x)}(t+) = f^{(x)}(t-)$.

Case A3: *PQ* rejects *p*, but *OPT* accepts *p*: *p* is an extra packet since only *OPT* accepts *p*. *p* is accepted into the $(h_{OPT}^{(x)}(t-) + 1)$st cell, which is free at *t*−, of $Q^{(x)}$. $h_{PQ}^{(x)}(t+) = h_{PQ}^{(x)}(t-) = B$, and $h_{OPT}^{(x)}(t+) = h_{OPT}^{(x)}(t-) + 1$, which means that $f^{(x)}(t+) = f^{(x)}(t-) - 1$.

Scheduling event:

If *PQ* (*OPT*, respectively) has at least one non-empty queue, suppose that *PQ* (*OPT*, respectively) transmits a packet from $Q^{(y)}$ ($Q^{(z)}$, respectively) at *t*.

Case S: $\sum_{j=1}^{m} h_{PQ}^{(j)}(t-) > 0$ and $\sum_{j=1}^{m} h_{OPT}^{(j)}(t-) > 0$:

Case S1: $y = z$:

Case S1.1: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) > 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-) - 1$, $h_{PQ}^{(y)}(t+) - h_{OPT}^{(y)}(t+) > 0$ holds. Thus, the $h_{OPT}^{(y)}(t-)$th cell of $Q^{(y)}$ becomes free in place of the $h_{PQ}^{(y)}(t-)$th cell of $Q^{(y)}$. Hence $f^{(y)}(t+) = f^{(y)}(t-)$.

Case S1.2: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) \le 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-) - 1$ hold, $h_{PQ}^{(y)}(t+) - h_{OPT}^{(y)}(t+) \le 0$. Hence the states of all the free cells do not change before and after *t*.

Case S2: $y > z$:

Case S2.1: $h_{PQ}^{(z)}(t-) - h_{OPT}^{(z)}(t-) < 0$:

Since $h_{PQ}^{(z)}(t+) = h_{PQ}^{(z)}(t-)$ and $h_{OPT}^{(z)}(t+) = h_{OPT}^{(z)}(t-) - 1$, $h_{PQ}^{(z)}(t+) \le h_{OPT}^{(z)}(t+)$. Thus, the states of all the free cells of $Q^{(z)}$ do not change before and after *t*.

Case S2.1.1: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) > 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-)$, $f^{(y)}(t+) = f^{(y)}(t-) - 1$ holds.

Case S2.1.2: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) \le 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-)$, $h_{PQ}^{(y)}(t+) < h_{OPT}^{(y)}(t+)$. Hence, the states of all the free cells of $Q^{(y)}$ do not change before and after *t*.

Please cite this article as: J. Kawahara et al., Tight analysis of priority queuing for egress traffic, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.09.001

Case S2.2: $h_{PQ}^{(z)}(t-) - h_{OPT}^{(z)}(t-) \geq 0$:

$h_{PQ}^{(z)}(t+) = h_{PQ}^{(z)}(t-)$ and $h_{OPT}^{(z)}(t+) = h_{OPT}^{(z)}(t-) - 1$. Thus, the $h_{OPT}^{(z)}(t-)$th cell of $Q^{(z)}$ becomes free, which means that $f^{(z)}(t+) = f^{(z)}(t-) + 1$ holds.

Case S2.2.1: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) > 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-)$, $f^{(y)}(t+) = f^{(y)}(t-) - 1$.

Case S2.2.2: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) \leq 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-)$, $h_{PQ}^{(y)}(t+) < h_{OPT}^{(y)}(t+)$, which means that the states of all the free cells of $Q^{(y)}$ do not change before and after t.

Case S3: $y < z$:

Since $h_{PQ}^{(z)}(t+) = h_{PQ}^{(z)}(t-) = 0$ by the definition of *PQ*, no new free cell arises in $Q^{(z)}$.

Case S3.1: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) > 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-)$, $f^{(y)}(t+) = f^{(y)}(t-) - 1$ holds.

Case S3.2: $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) \leq 0$:

Since $h_{PQ}^{(y)}(t+) = h_{PQ}^{(y)}(t-) - 1$ and $h_{OPT}^{(y)}(t+) = h_{OPT}^{(y)}(t-)$, $h_{PQ}^{(y)}(t+) < h_{OPT}^{(y)}(t+)$ holds. Hence, the states of all the free cells of $Q^{(y)}$ do not change before and after t.

Case $\bar{S}$: $\sum_{j=1}^{m} h_{PQ}^{(j)}(t-) = 0$ and $\sum_{j=1}^{m} h_{OPT}^{(j)}(t-) > 0$:

Since the buffers of *PQ* are empty, there does not exist any free cell in them.

Based on a change in the state of free cells, we match each extra packet with a packet transmitted by *PQ* according to the matching routine in Table 1. (All the names of the cases in the routine correspond to the names of cases in the above sketch about free cells.) We outline the matching routine. Roughly speaking, the routine either adds a new edge to a tentative matching if a new free cell arises (Cases A1, S1.1, S2.2), or fixes some edge if *OPT* accepts an extra packet (Case A3), while keeping edges constructed before. In the other cases (Cases A2, S1.2, S2.1, S3, $\bar{S}$), the routine does nothing. Specifically, both *OPT* and *PQ* accept arriving packets at the same queue in Case A1, and they transmit packets from the same queue in Case S1.1. Since the total numbers of free cells do not change in these cases but the states of free cells do, the routine updates an edge in a tentative matching, namely removes an edge between *PQ*'s packet p and a cell that became non-free and adds a new edge between p and a new free cell. When the routine executes Case S2.2, the queue where *OPT* transmits a packet is different from that of *PQ*. By the conditions of the numbers of packets in their queues and so on (see the condition of Case S2.2), a cell of *OPT*'s queue becomes free. The routine matches the cell with the packet transmitted by *PQ* at this event. In Case A3, an extra packet is accepted into a free cell c. Since c has been already matched with some *PQ*'s packet p', which can be proven inductively in Lemma 3.3, the routine replaces the partner of p' from c to p. Once an extra packet is matched, the partner of the packet never changes.

We give some definitions. For any packet p, $g(p)$ denotes the index of the queue at which p arrives. Also, for any cell c, $g(c)$ denotes the index of the queue including c. We now show the feasibility of the routine.

Lemma 3.3. *For any non-event time t', and any extra packet p which arrives before t', there exists some packet p' such that PQ transmits p' before t', $g(p) < g(p')$ and p is matched with p' at t'. Moreover, for any free cell c at t', there exists some packet p'' such that PQ transmits p'' before t', $g(c) < g(p'')$, and c is matched with p'' at t'.*

Proof. The proof is by induction on the event time. The base case is clear. Let t be any event time. We assume that the statement is true at $t-$, and prove that it is true at $t+$.

First, we discuss the case where the routine executes Case A1 or S1.1 at t. Let c be the cell which becomes free at t. Also, let c' be the cell which is free at $t-$ and not free at $t+$. By the induction hypothesis, a packet p which is transmitted by *PQ* before $t-$ is matched with c' at $t-$. Then, the routine unmatches p, and matches p with c by the definitions of Cases A1 and S1.1. $g(c) = g(c')$ clearly holds. Also, since $g(c') < g(p)$ by the induction hypothesis, the statement is true at $t+$.

Table 1
Matching routine.

Matching routine: Let t be an event time.
Arrival event: Suppose that the packet p arrives at $Q^{(x)}$ at t. Execute one of the following three cases at t.
Case A1: Both *PQ* and *OPT* accept p, and $h_{PQ}^{(x)}(t-) - h_{OPT}^{(x)}(t-) > 0$:
Let c be *OPT*'s $(h_{OPT}^{(x)}(t-) + 1)$st cell of $Q^{(x)}$, which is free at $t-$ but not at $t+$. Let c' be *OPT*'s $(h_{PQ}^{(x)}(t-) + 1)$st cell which is not free at $t-$ but is free at $t+$. There exists the packet q matched with c at $t-$. (The existence of such q is guaranteed by Lemma 3.3.) Change the matching partner of q from c to c'.
Case A2: Both *PQ* and *OPT* accept p, and $h_{PQ}^{(x)}(t-) - h_{OPT}^{(x)}(t-) \leq 0$:
Do nothing.
Case A3: *PQ* rejects p, but *OPT* accepts p:
Let c be *OPT*'s $(h_{OPT}^{(x)}(t-) + 1)$st cell of $Q^{(x)}$, that is, the cell to which the extra packet p is now stored. Note that c is free at $t-$ but is not at $t+$. There exists the packet q matched with c at $t-$. (See Lemma 3.3.) Change the partner of q from c to p.
Scheduling event: If *PQ* (*OPT*, respectively) has at least one non-empty queue at $t-$, suppose that *PQ* (*OPT*, respectively) transmits a packet from $Q^{(y)}$ ($Q^{(z)}$, respectively) at t. Execute one of the following three cases at t.
Case S1.1: $\sum_{j=1}^{m} h_{PQ}^{(j)}(t-) > 0$, $\sum_{j=1}^{m} h_{OPT}^{(j)}(t-) > 0$, $y = z$, and $h_{PQ}^{(y)}(t-) - h_{OPT}^{(y)}(t-) > 0$:
Let c be *OPT*'s $h_{PQ}^{(y)}(t-)$th cell of $Q^{(y)}$, which is free at $t-$ but is not free at $t+$. Let c' be *OPT*'s $h_{OPT}^{(y)}(t-)$th cell of $Q^{(y)}$, which is not free at $t-$ but is free at $t+$. There exists the packet q matched with c at $t-$. (See Lemma 3.3.) Change the matching partner of q from c to c'.
Case S2.2: $\sum_{j=1}^{m} h_{PQ}^{(j)}(t-) > 0$, $\sum_{j=1}^{m} h_{OPT}^{(j)}(t-) > 0$, $y > z$, and $h_{PQ}^{(z)}(t-) - h_{OPT}^{(z)}(t-) \geq 0$:
Let c be *OPT*'s $h_{OPT}^{(z)}(t-)$th cell of $Q^{(z)}$, which becomes free at $t+$. Since the packet p transmitted from $Q^{(y)}$ by *PQ* is not matched with anything (see Lemma 3.3), match p with c.
Otherwise (Cases S1.2, S2.1, S3, $\bar{S}$): Do nothing.

Next, we consider the case where the routine executes Case A3 at t. Let p' be the extra packet accepted by *OPT* at t. Also, let c be the free cell into which *OPT* accepts p' at t. By the induction hypothesis, a packet p which is transmitted by *PQ* before $t-$ is matched with c at $t-$. Then, by the definition of Case A3, the routine unmatches p, and matches p with p'. $g(c) = g(p')$ holds by definition. In addition, $g(c) < g(p)$ by the induction hypothesis. Thus, $g(p') < g(p)$, which means that the statement holds at $t+$.

Third, we investigate the case where the routine executes Case S2.2 at t. Suppose that *PQ* transmits a packet p at t, and the new free cell c arises at t. By the induction hypothesis, any *PQ*'s packet which is matched with a free cell or an extra packet at $t-$ is transmitted before t. Hence, p is not matched with anything at $t-$. Thus, the routine can match p with c at t. Moreover, $g(c) < g(p)$ by the condition of Case S2.2. By the induction hypothesis, the statement is true at $t+$.

In the other cases, a new matching does not arise. Therefore, the statement is clear by the induction hypothesis, which completes the proof. □

In the next lemma, we obtain part of the properties of the set $\mathcal{S}^*$.

Lemma 3.4. *Let σ be an input such that for some $u(\in [1, m])$, $s_u(\sigma) > B$. Then, there exists an input $\hat{\sigma}$ such that*

for each $j(\in [1, m])$, $s_j(\hat{\sigma}) \le B$, and $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} < \frac{V_{OPT}(\hat{\sigma})}{V_{PQ}(\hat{\sigma})}$.

Proof. Let z be the minimum index such that $s_z(\sigma) > B$. Then, there exist the three event times t_1, $t_2(> t_1)$ and $t_3(> t_2)$ satisfying the following three conditions: (i) t_2 is the arrival event time when the $(B+1)$st packet which *PQ* accepts at $Q^{(z)}$ arrives, (ii) *OPT* does not transmit any packet from $Q^{(z)}$ during time (t_1, t_2), where t_1 is the event time when *OPT* transmits a packet from $Q^{(z)}$, (Since *OPT* accepts any arriving packet by our assumption, *OPT* certainly transmits at least one packet from $Q^{(z)}$ before t_2.) and (iii) *PQ* does not transmit any packet from $Q^{(z)}$ during time (t_2, t_3), where t_3 is the event time when *PQ* transmits a packet from $Q^{(z)}$. We construct σ' by removing the events at t_1 and t_2 from σ. Suppose that $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} < \frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')}$. If we remove some events corresponding to $Q^{(j)}$ in ascending order of index j in $\{x | s_x(\sigma) > B\}$, then we can construct an input $\hat{\sigma}$ such that for each $j(\in [1, m])$, $s_j(\hat{\sigma}) \le B$, and $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} < \frac{V_{OPT}(\hat{\sigma})}{V_{PQ}(\hat{\sigma})}$, which completes the proof. Hence, we next show that $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} < \frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')}$.

First, we discuss the gain of *OPT* for σ'. Let *ALG* be the offline algorithm for σ' such that for each scheduling event e in σ', *ALG* selects the queue which *OPT* selects at e in σ. We consider the number of packets in *ALG*'s buffer during time (t_1, t_3) for σ'. For any non-event time $t(\in (t_1, t_3))$, and any $y(\neq z)$, $h^{(y)}_{ALG}(t) = h^{(y)}_{OPT}(t)$. For any non-event time $t(\in (t_1, t_2))$, $h^{(z)}_{ALG}(t) = h^{(z)}_{OPT}(t) + 1$. Also, for any non-event time $t(\in (t_2, t_3))$, $h^{(z)}_{ALG}(t) = h^{(z)}_{OPT}(t)$. By the above argument, $V_{OPT}(\sigma') \ge V_{ALG}(\sigma') = V_{OPT}(\sigma) - \alpha_z$.

Next, we evaluate the gain of *PQ* for σ'. For notational simplicity, we describe *PQ* for σ' as *PQ'*. First, we consider the case where there does not exist any packet which *PQ* accepts but *PQ'* rejects during time (t_1, t_3). To evaluate the gain of *PQ'* in this case, we discuss the numbers of packets which *PQ* and *PQ'* store in their buffers after t_1. For any non-event time $t(\in (t_1, t_2))$, $\sum_{j=1}^{m} h^{(j)}_{PQ'}(t) = \sum_{j=1}^{m} h^{(j)}_{PQ}(t) + 1$. For any non-event time $\hat{t}$, we define $w(\hat{t}) = \arg\max\{j \mid h^{(j)}_{PQ'}(\hat{t}) > 0\}$. Specifically, $h^{(w(t))}_{PQ'}(t) = h^{(w(t))}_{PQ}(t) + 1$. (We call this fact the property (a).) Moreover, for any non-event time $t(\in (t_2, t_3))$, $\sum_{j=1}^{m} h^{(j)}_{PQ'}(t) = \sum_{j=1}^{m} h^{(j)}_{PQ}(t)$. However, if $w(t) > z$, then $h^{(w(t))}_{PQ'}(t) = h^{(w(t))}_{PQ}(t) + 1$. Also, $h^{(z)}_{PQ'}(t) = h^{(z)}_{PQ}(t) - 1$. If $w(t) = z$, then for any $j(\in [1, m])$, $h^{(j)}_{PQ'}(t) = h^{(j)}_{PQ}(t)$. For any non-event time $t(> t_3)$ and any $j(\in [1, m])$, $h^{(j)}_{PQ'}(t) = h^{(j)}_{PQ}(t)$. By the above argument, $V_{PQ}(\sigma') = V_{PQ}(\sigma) - \alpha_z$ holds.

Secondly, we consider the case where there exists at least one packet which *PQ* accepts but *PQ'* rejects. Let t' be the first event time when the packet p which *PQ* accepts but *PQ'* rejects arrives. Then, suppose that $t' \in (t_1, t_2)$. By the definition of z, p arrives at $Q^{(z')}$ such that $z' \ge z$. By the property (a), for $j(\in [1, m])$, $h^{(j)}_{PQ'}(t'+) = h^{(j)}_{PQ}(t'+)$. Thus, packets accepted by *PQ* during time (t', t_2) can be accepted by *PQ'*. Only *PQ* accepts the packet arriving at $Q^{(z)}$ at t_2 by the definition of σ'. Hence, $h^{(z)}_{PQ'}(t_2+) = h^{(z)}_{PQ}(t_2+) - 1$, and for any $j(\in [1, m])$ such that $j \neq z$, $h^{(j)}_{PQ'}(t_2+) = h^{(j)}_{PQ}(t_2+)$. (We call this fact the property (b).) If all the packets which *PQ* accepts after t_2 are the same as those accepted by *PQ'* after t_2, $V_{PQ}(\sigma') = V_{PQ}(\sigma) - \alpha_z - \alpha_{z'}$. Then, we consider the case where there exists at least one packet p' which *PQ* rejects but *PQ'* accepts after t_2. By the greediness of *PQ* and the property (b), for any non-event time $t(> t_2)$ and any $y'(\ge z+1)$, $h^{(y')}_{PQ'}(t) = h^{(y')}_{PQ}(t)$. Hence, p' arrives at $Q^{(z'')}$ for some $z''(\le z)$. Let t'' be the event time when p' arrives. For any $j(\in [1, m])$, $h^{(j)}_{PQ'}(t''+) = h^{(j)}_{PQ}(t''+)$, which means that all the packets accepted by *PQ* are equal to those accepted by *PQ'* after t''. Thus, $V_{PQ}(\sigma') = V_{PQ}(\sigma) - \alpha_z - \alpha_{z'} + \alpha_{z''} \le V_{PQ}(\sigma) - \alpha_z$.

Finally, we consider the case where $t' \in (t_2, t_3)$. By the same argument as the case of $t' \in (t_1, t_2)$, we can prove this case. Specifically, the number of packets which *PQ* rejects but *PQ'* accepts after t' is exactly one. This packet arrives at $Q^{(z''')}$, where some $z''' \le z$. Therefore, $V_{PQ}(\sigma') = V_{PQ}(\sigma) - \alpha_z - \alpha_{z'} + \alpha_{z'''} \le V_{PQ}(\sigma) - \alpha_z$.

By the above argument, $\frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')} \ge \frac{V_{ALG}(\sigma')}{V_{PQ}(\sigma')} \ge \frac{V_{OPT}(\sigma)-\alpha_z}{V_{PQ}(\sigma)-\alpha_z} > \frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)}$. □

We give the notation. $\mathcal{S}_1$ denotes the set of inputs σ such that for any $j(\in [1, m])$, $s_j(\sigma) \le B$. In what follows, we analyze only inputs in $\mathcal{S}_1$ by Lemma 3.4. Next, we evaluate the number of extra packets arriving at each good queue using Lemma 3.3.

Lemma 3.5. *For any $x(\in [1, n])$, $\sum_{i=x}^{n} k_{q_i} \le \sum_{j=q_x+1}^{m} s_j$.*

Proof. By Lemma 3.3, each extra packet p is matched with a packet p' transmitted by *PQ* at the end of the input. In addition, $g(p) < g(p')$ if an extra packet p is matched with a packet p' of *PQ*. Thus, $k_{q_n} \le \sum_{j=q_n+1}^{m} s_j$, $k_{q_{n-1}} \le (\sum_{j=q_{n-1}+1}^{m} s_j) - k_{q_n}$, $\cdots$, and $k_{q_1} \le (\sum_{j=q_1+1}^{m} s_j) - \sum_{i=2}^{n} k_{q_i}$. Therefore, for any $x(\in [1, n])$, $\sum_{i=x}^{n} k_{q_i} \le \sum_{j=q_x+1}^{m} s_j$. □

Now we gradually gain all the properties of $\mathcal{S}^*$ in the following lemmas while proving $\mathcal{S}^*$ contains inputs σ such that

$\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)}$ is maximized. Specifically, for $i = 1, \ldots, 4$, we construct some subset $\mathcal{S}_{i+1}$ from the set $\mathcal{S}_i$ in each of the following lemmas, and eventually we can gain $\mathcal{S}^*$ from $\mathcal{S}_5$. (We have already obtained $\mathcal{S}_1$ in Lemma 3.4.) It is difficult to show all the properties of $\mathcal{S}^*$ in one lemma, and thus we progressively give the definitions of the $\mathcal{S}_{i+1}$ that has more restrictive properties than $\mathcal{S}_i$.

Next in Lemma 3.6, we discuss the condition of events where the number of extra packets accepted into a good queue $Q^{(q_i)}$ $(i \in [1, n])$ is maximized, and show that it is true when $k_{q_i} = \sum_{j=q_i+1}^{q_{i+1}} s_j$. Throughout the proofs of all the following lemmas, we drop σ from $s_j(\sigma)$, $n(\sigma)$, $q_i(\sigma)$ and $k_j(\sigma)$.

Lemma 3.6. *For any input $\sigma \in \mathcal{S}_1$, there exists an input $\hat{\sigma}(\in \mathcal{S}_1)$ such that (i) for any $i(\in [1, n(\hat{\sigma})])$, $k_{q_i(\hat{\sigma})}(\hat{\sigma}) = \sum_{j=q_i(\hat{\sigma})+1}^{q_{i+1}(\hat{\sigma})} s_j(\hat{\sigma})$, (ii) for any $j(\in [1, q_1(\hat{\sigma}) - 1])$, $s_j(\hat{\sigma}) = 0$ if $q_1(\hat{\sigma}) - 1 \geq 1$, and (iii) $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \leq \frac{V_{OPT}(\hat{\sigma})}{V_{PQ}(\hat{\sigma})}$.*

Proof. For any input $\sigma \in \mathcal{S}_1$, we construct σ' from σ according to the following steps. First, for each $j(\in [q_1, m])$, s_j events at which s_j packets arrive at $Q^{(j)}$ occur during time $(0, 1)$. Since $s_j \leq B$ by the definition of $\mathcal{S}_1$, *PQ* accepts all the packets which arrive at these events. $\sum_{i=1}^{n} k_{q_i}$ packets arrive after time 1, and *PQ* cannot accept them. Specifically, for any $i(\in [1, n])$, we define $a_i = \sum_{j=q_{n+1-i}+1}^{q_{n+2-i}} s_j$ and $a_0 = 0$. Then, for each $x(\in [0, n-1])$, a scheduling event occurs at each integer time $t = (\sum_{j=0}^{x} a_j) + 1, \ldots, \sum_{j=0}^{x+1} a_j$, and an arrival event where a packet arrives at $Q^{(q_{n-x})}$ occurs at each time $t + \frac{1}{2}$. After time $(\sum_{j=0}^{n} a_j) + 1$, sufficient scheduling events to transmit all the arriving packets occur.

For these scheduling events, *PQ* transmits a packet from $Q^{(j)}$ at t, where j is an integer between $q_{n-x} + 1$ and q_{n-x+1}. Also, let *ALG* be an offline algorithm. *ALG* transmits a packet from $Q^{(q_{n-x})}$ at t. Since for any $i(\in [1, n])$, at least one extra packet arrives at $Q^{(q_i)}$, $s_{q_i} = B$ holds. Hence, since for any $i(\in [1, n])$, $h_{PQ}^{(q_i)}(1-) = B$, *PQ* cannot accept the packet which arrives at each $t + \frac{1}{2}$. However, *ALG* can accept all these packets, which means that *ALG* is an optimal offline algorithm. Then, $n(\sigma') = n$, and for any $i(\in [1, n])$, $q_i(\sigma') = q_i$.

By the above argument, $V_{PQ}(\sigma') = V_{PQ}(\sigma) - \sum_{j=1}^{q_1-1} \alpha_j s_j$. Furthermore, for each $i(\in [1, n])$, $k_{q_i}(\sigma') = \sum_{j=q_i+1}^{q_{i+1}} s_j$. By these equalities, $V_{ALG}(\sigma') = V_{PQ}(\sigma') + \sum_{i=1}^{n} \alpha_{q_i} k_{q_i}(\sigma') = V_{PQ}(\sigma) + \sum_{i=1}^{n} \alpha_{q_i} (\sum_{j=q_i+1}^{q_{i+1}} s_j) - \sum_{j=1}^{q_1-1} \alpha_j s_j = V_{PQ}(\sigma) + \alpha_{q_1} (\sum_{j=q_1+1}^{q_{n+1}} s_j) + \sum_{x=2}^{n} (\alpha_{q_x} - \alpha_{q_{x-1}})(\sum_{j=q_x+1}^{q_{n+1}} s_j) - \sum_{j=1}^{q_1-1} \alpha_j s_j$. Since $\sum_{i=x}^{n} k_{q_i} \leq \sum_{j=q_x+1}^{m} s_j$ by Lemma 3.5 and $q_{n+1} = m$, $V_{ALG}(\sigma') \geq V_{PQ}(\sigma) + \alpha_{q_1} (\sum_{i=1}^{n} k_{q_i}) + \sum_{x=2}^{n} (\alpha_{q_x} - \alpha_{q_{x-1}})(\sum_{i=x}^{n} k_{q_i}) - \sum_{j=1}^{q_1-1} \alpha_j s_j = V_{PQ}(\sigma) + \sum_{i=1}^{n} \alpha_{q_i} k_{q_i} - \sum_{j=1}^{q_1-1} \alpha_j s_j = V_{OPT}(\sigma) - \sum_{j=1}^{q_1-1} \alpha_j s_j$.

Therefore, $\frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')} = \frac{V_{ALG}(\sigma')}{V_{PQ}(\sigma')} \geq \frac{V_{OPT}(\sigma) - \sum_{j=1}^{q_1-1} \alpha_j s_j}{V_{PQ}(\sigma) - \sum_{j=1}^{q_1-1} \alpha_j s_j} \geq \frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)}$. Moreover, by the definition of σ', σ' satisfies the condition (ii) in the statement, which means that $\mathcal{S}_1$ includes σ'. □

In light of the above lemma, we introduce the next set of inputs. $\mathcal{S}_2$ denotes the set of inputs $\sigma(\in \mathcal{S}_1)$ satisfying the following conditions: (i) for any $i(\in [1, n])$, $k_{q_i} = \sum_{j=q_i+1}^{q_{i+1}} s_j$, (ii) for any $j(\in [q_1, m])$, $s_j \leq B$, and (iii) for any $j(\in [1, q_1 - 1])$, $s_j = 0$ if $q_1 - 1 \geq 1$.

Lemma 3.7. *Let $\sigma(\in \mathcal{S}_2)$ be an input such that for some $z(\leq n(\sigma) - 1)$, $q_z(\sigma) + 1 < q_{z+1}(\sigma)$. Then, there exists an input $\hat{\sigma}(\in \mathcal{S}_2)$ such that (i) for each $i(\in [1, n(\hat{\sigma}) - 1])$, $q_i(\hat{\sigma}) + 1 = q_{i+1}(\hat{\sigma})$ and $k_{q_i(\hat{\sigma})}(\hat{\sigma}) = B$, and (ii) $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \leq \frac{V_{OPT}(\hat{\sigma})}{V_{PQ}(\hat{\sigma})}$.*

Proof. For any $j(\in [1, m])$ such that $j \neq q_{z+1} - 1$, we define $s'_j = s_j$. Also, we define $s'_{q_{z+1}-1} = B$. (See Fig. 1.)

We construct σ' from σ in the following way. This approach is similar to those in the proof of Lemma 3.6. First, for each $j(\in [q_1, m])$, s'_j events at which s'_j packets arrive at $Q^{(j)}$ occur during time $(0, 1)$. Since $s'_j \leq B$ by definition, *PQ* accepts all these packets. In addition, for any $i(\in [1, z])$, we define $q'_i = q_i$. We define $q'_{z+1} = q_{z+1} - 1$. For any $i(\in [z+1, n+1])$, we define $q'_{i+1} = q_i$. Moreover, for any $i(\in [1, n+1])$, we define $a_i = \sum_{j=q'_{n+2-i}+1}^{q'_{n+3-i}} s'_j$ and

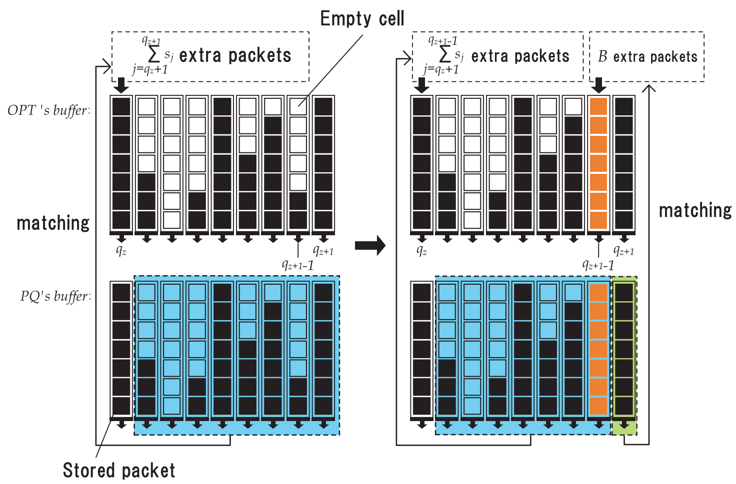

Fig. 1. Example states of queues (q_z through q_{z+1}) of *OPT* and *PQ* for σ and σ'. Left (Right) queues show the states for σ (σ').

Please cite this article as: J. Kawahara et al., Tight analysis of priority queuing for egress traffic, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.09.001

$a_0 = 0$. For any $x(\in [0, n])$, a scheduling event occurs at each integer time $t = (\sum_{j=0}^{x} a_j) + 1, \ldots, \sum_{j=0}^{x+1} a_j$. Also, an arrival event where a packet arrives at $Q^{(q'_{n-x+1})}$ occurs at each time $t + \frac{1}{2}$. After time $(\sum_{j=0}^{n+1} a_j) + 1$, sufficient scheduling events to transmit all the arriving packets occur.

Then, *PQ* transmits a packet from $Q^{(j)}$ at t, where j is an integer between $q'_{n-x+1} + 1$ and q'_{n-x+2}. Let *ALG* be an offline algorithm which transmits a packet from $Q^{(q'_{n-x+1})}$ at t. By the definition of q'_i, for any $i(\in [1, n+1])$, $h_{PQ}^{(q'_i)}(1-) = B$. Thus, *PQ* cannot accept any packet arriving at $t + \frac{1}{2}$, but *ALG* can accept all the arriving packets. That is to say, *ALG* is optimal.

By the above argument, $V_{PQ}(\sigma') = V_{PQ}(\sigma) + \alpha_{q_{z+1}-1}(B - s_{q_{z+1}-1})$. Furthermore, for any $j(\neq q_z, q_{z+1}-1)$, $k_j(\sigma') = k_j$. Also, $k_{q_z}(\sigma') = k_{q_z} - s_{q_{z+1}-1}$ and $k_{q_{z+1}-1}(\sigma') = B$. Also, for any $i(\in [1, n+1])$, $q_i(\sigma') = q'_i$. Moreover, $V_{OPT}(\sigma') = V_{ALG}(\sigma') = V_{PQ}(\sigma') + \sum_{i=1}^{n(\sigma')} \alpha_{q_i(\sigma')} k_{q_i(\sigma')}(\sigma')$.

By the above equalities, $\sum_{i=1}^{n(\sigma')} \alpha_{q_i(\sigma')} k_{q_i(\sigma')}(\sigma') = (\sum_{i=1}^{n} \alpha_{q_i} k_{q_i}) - \alpha_{q_z} s_{q_{z+1}-1} + \alpha_{q_{z+1}-1} B \geq (\sum_{i=1}^{n} \alpha_{q_i} k_{q_i}) + \alpha_{q_{z+1}-1}(B - s_{q_{z+1}-1})$. Hence, $\frac{\sum_{i=1}^{n(\sigma')} \alpha_{q_i(\sigma')} k_{q_i(\sigma')}(\sigma')}{V_{PQ}(\sigma')} \geq \frac{(\sum_{i=1}^{n} \alpha_{q_i} k_{q_i}) + \alpha_{q_{z+1}-1}(B - s_{q_{z+1}-1})}{V_{PQ}(\sigma) + \alpha_{q_{z+1}-1}(B - s_{q_{z+1}-1})} \geq \frac{\sum_{i=1}^{n} \alpha_{q_i} k_{q_i}}{V_{PQ}(\sigma)}$. Therefore, $\frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')} \geq \frac{V_{PQ}(\sigma') + \sum_{i=1}^{n(\sigma')} \alpha_{q_i(\sigma')} k_{q_i(\sigma')}(\sigma')}{V_{PQ}(\sigma')} \geq 1 + \frac{\sum_{i=1}^{n} \alpha_{q_i} k_{q_i}}{V_{PQ}(\sigma)} = \frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)}$.

By the definition of σ', $\mathcal{S}_2$ includes σ'. By the above argument, for any z' such that $q_{z'} + 1 < q_{z'+1}$, we recursively construct an input in the above way, and then we can obtain an input satisfying the lemma. □

We define the set $\mathcal{S}_3$ of inputs. $\mathcal{S}_3$ denotes the set of inputs $\sigma(\in \mathcal{S}_2)$ such that (i) for each $i(\in [1, n-1])$, $q_i + 1 = q_{i+1}$, (ii) for each $i(\in [1, n-1])$, $k_{q_i} = B$, (iii) for each $j(\in [q_1, q_n])$, $s_j = B$, (iv) for any $j(\in [1, q_1 - 1])$, $s_j = 0$ if $q_1 - 1 \geq 1$, and (v) for each $j(\in [q_n + 1, m])$, $s_j \leq B$. (By Lemma 3.2, $q_n + 1 \leq m$.)

Lemma 3.8. *For any input $\sigma(\in \mathcal{S}_3)$, there exists an input $\sigma'(\in \mathcal{S}_3)$ such that (i) $s_{q_{n(\sigma)}(\sigma)+u+1}(\sigma') = (\sum_{j=q_{n(\sigma)}(\sigma)+1}^{m} s_j(\sigma)) - uB$, where $u = \lfloor \frac{\sum_{j=q_{n(\sigma)}(\sigma)+1}^{m} s_j(\sigma)}{B} \rfloor$, and for any $j(\in [q_{n(\sigma)}(\sigma), q_{n(\sigma)}(\sigma) + u])$, $s_j(\sigma') = B$, and (ii) $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \leq \frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')}$.*

Proof. For any $j(\in [1, q_n])$, we define $s''_j = s_j$. Furthermore, for each $j(\in [q_n + 1, q_n + u])$, we define $s''_j = B$, and $s''_{q_n+u+1} = (\sum_{j=q_n+1}^{m} s_j) - uB$. Also, for each $j(\in [q_n + u + 2, m])$, we define $s''_j = 0$ if $q_n + u + 2 \leq m$.

We construct σ' from σ in the following way. This approach is similar to those in the proof of Lemmas 3.6 and 3.7. First, for each $j(\in [q_1, m])$, s''_j events at which s''_j packets arrive at $Q^{(j)}$ occur during time $(0, 1)$. Since $s''_j \leq B$ by definition, *PQ* accepts all these packets. Then, for any $i(\in [1, n])$, we define $a_i = \sum_{j=q_{n+1-i}+1}^{q_{n+2-i}} s_j$, and $a_0 = 0$. For any $x(\in [0, n-1])$, a scheduling event occurs at each integer time $t = (\sum_{j=0}^{x} a_j) + 1, \ldots, \sum_{j=0}^{x+1} a_j$. Also, at each time $t + \frac{1}{2}$, an arrival event where a packet arrives at $Q^{(q_{n-x})}$ occurs. After time $(\sum_{j=0}^{n} a_j) + 1$, sufficient scheduling events to transmit all the arriving packets occur. $V_{PQ}(\sigma') = V_{PQ}(\sigma) - \sum_{j=q_n+1}^{m} \alpha_j s_j + \sum_{j=q_n+1}^{q_n+u+1} \alpha_j s''_j$ and $V_{OPT}(\sigma') = V_{OPT}(\sigma) - \sum_{j=q_n+1}^{m} \alpha_j s_j + \sum_{j=q_n+1}^{q_n+u+1} \alpha_j s''_j$. Since $-\sum_{j=q_n+1}^{m} \alpha_j s_j + \sum_{j=q_n+1}^{q_n+u+1} \alpha_j s''_j \leq 0$ by definition, $\frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')} = \frac{V_{OPT}(\sigma) - \sum_{j=q_n+1}^{m} \alpha_j s_j + \sum_{j=q_n+1}^{q_n+u+1} \alpha_j s''_j}{V_{PQ}(\sigma) - \sum_{j=q_n+1}^{m} \alpha_j s_j + \sum_{j=q_n+1}^{q_n+u+1} \alpha_j s''_j} \geq \frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)}$. Moreover, by the definition of σ', $\sigma' \in \mathcal{S}_3$ holds, and σ' satisfies the condition (i) in the statement. □

We next introduce the set $\mathcal{S}_4$ of inputs. Let $\mathcal{S}_4$ denote the set of inputs $\sigma(\in \mathcal{S}_3)$ satisfying the following five conditions: (i) for each $i(\in [1, n-1])$, $q_i + 1 = q_{i+1}$, (ii) for each $i(\in [1, n-1])$, $k_{q_i} = B$, (iii) for each $j(\in [q_1, q_n])$, $s_j = B$, (iv) for any $j(\in [1, q_1 - 1])$, $s_j = 0$ if $q_1 - 1 \geq 1$, and (v) there exists some u such that $0 \leq u \leq m - q_n - 1$. Also, for any $j(\in [q_n, q_n + u])$, $s_j = B$, $B \geq s_{q_n+u+1} \geq 1$, and for any $j(\in [q_n + u + 2, m])$, $s_j = 0$ if $q_n + u + 2 \leq m$.

Lemma 3.9. *Let $\sigma(\in \mathcal{S}_4)$ be an input such that $q_{n(\sigma)}(\sigma) + 2 \leq m$, $s_{q_{n(\sigma)}(\sigma)+1}(\sigma) = B$, and $\sum_{j=q_{n(\sigma)}(\sigma)+2}^{m} s_j(\sigma) > 0$.*

Then, there exists an input $\hat{\sigma}(\in \mathcal{S}_4)$ such that (i) $n(\hat{\sigma}) = n(\sigma) + 1$, (ii) for each $i(\in [1, n(\hat{\sigma}) - 1])$, $q_i(\hat{\sigma}) = q_i(\sigma)$, and $q_{n(\hat{\sigma})}(\hat{\sigma}) = q_{n(\sigma)}(\sigma) + 1$, and (iii) $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \leq \frac{V_{OPT}(\hat{\sigma})}{V_{PQ}(\hat{\sigma})}$.

Proof. We construct σ' from σ as follows: First, for each $j(\in [q_1, m])$, s_j events at which s_j packets at $Q^{(j)}$ arrive occur during time $(0, 1)$. Since $s_j \leq B$ by the definition of $\mathcal{S}_4$, *PQ* accepts all these arriving packets. For any $i(\in [1, n])$, we define $q'_i = q_i$, $q'_{n+1} = q_n + 1$ and $q'_{n+2} = m$. Moreover, for any $i(\in [1, n+1])$, we define $a_i = \sum_{j=q'_{n+2-i}+1}^{q'_{n+3-i}} s_j$ and $a_0 = 0$. Then, for any $x(\in [0, n])$, a scheduling event occurs at each integer time $t = (\sum_{j=0}^{x} a_j) + 1, \ldots, \sum_{j=0}^{x+1} a_j$. In addition, for any $x(\in [0, n])$, an arrival event where a packet arrives at $Q^{(q'_{n+1-x})}$ occurs at each time $t + \frac{1}{2}$. After time $(\sum_{j=0}^{n+1} a_j) + 1$, sufficient scheduling events to transmit all the arriving packets occur.

Then, the packets which *PQ* transmits at each scheduling event for σ' are equivalent to those for σ. Consider an offline algorithm *ALG* which transmits a packet from $Q^{(q'_{n+1-x})}$ at t. By the definition of q'_i, since for any $i(\in [1, n+1])$, $h_{PQ}^{(q'_i)}(1-) = B$, *PQ* cannot accept any packet which arrives at each time $t + \frac{1}{2}$, but *ALG* can accept all the packets, which means that *ALG* is optimal. Hence, $n(\sigma') = n + 1$, and for any $i(\in [1, n+1])$, $q_i(\sigma') = q'_i$.

Since for any $j(\in [1, m])$, $s_j(\sigma') = s_j$, $V_{PQ}(\sigma') = V_{PQ}(\sigma)$. Moreover, for any $i(\in [1, n-1])$, $k_{q_i}(\sigma') = k_{q_i}$, $k_{q_n}(\sigma') = s_{q_n+1}$, and $k_{q_n+1}(\sigma') = \sum_{j=q_n+2}^{m} s_j$. Therefore, $\sigma' \in \mathcal{S}_4$ holds, and σ' satisfies the conditions (i) and (ii) in the statements. Also, $V_{OPT}(\sigma') = V_{ALG}(\sigma') = V_{OPT}(\sigma) + (\alpha_{q_n+1} - \alpha_{q_n}) \sum_{j=q_n+2}^{m} s_j \geq V_{OPT}(\sigma)$. □

$\mathcal{S}_5$ denotes the set of inputs $\sigma(\in \mathcal{S}_4)$ satisfying the following six conditions: (i) for each $i(\in [1, n-1])$, $q_i + 1 = q_{i+1}$, (ii) for each $i(\in [1, n-1])$, $k_{q_i} = B$, (iii) for each $j(\in [q_1, q_n])$, $s_j = B$, (iv) for any $j(\in [1, q_1 - 1])$, $s_j = 0$ holds if $q_1 - 1 \geq 1$, (v) $k_{q_n} = s_{q_n+1}$ (By Lemma 3.2, $q_n + 1 \leq m$.) and

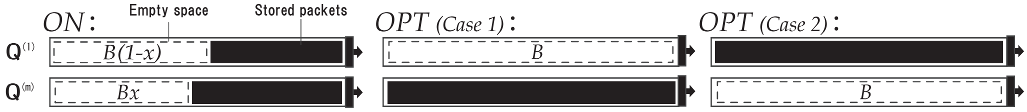

Fig. 2. States of queues at time 2.

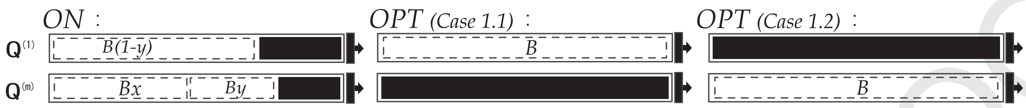

Fig. 3. States of queues at time 4 via Case 1.

$1 \le s_{q_n+1} \le B$, and (vi) for any $j(\in[q_n+2, m])$, $s_j = 0$ holds if $q_n + 2 \le m$.

Lemma 3.10. *For any input $\sigma(\in \mathcal{S}_5)$, there exists an input $\hat{\sigma}(\in \mathcal{S}_5)$ such that (i) $s_{q_{n(\hat{\sigma})}(\hat{\sigma})+1}(\hat{\sigma}) = B$, and (ii) $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \le \frac{V_{OPT}(\hat{\sigma})}{V_{PQ}(\hat{\sigma})}$.*

That is, there exists an input $\sigma^* \in \mathcal{S}^*$ such that $\max_{\sigma'}\{\frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')}\} = \frac{V_{OPT}(\sigma^*)}{V_{PQ}(\sigma^*)}$.

Proof. Since $\sigma \in \mathcal{S}_5$ holds, $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} = \frac{V_{PQ}(\sigma)+\sum_{i=1}^{n}\alpha_{q_i}k_{q_i}}{V_{PQ}(\sigma)} \le 1 + \frac{B(\sum_{j=q_1}^{q_n-1}\alpha_j)+\alpha_{q_n}s_{q_n+1}}{\sum_{j=q_1}^{q_n+1}\alpha_j s_j} \le 1 + \frac{B(\sum_{j=q_1}^{q_n-1}\alpha_j)+\alpha_{q_n}s_{q_n+1}}{B(\sum_{j=q_1}^{q_n}\alpha_j)+\alpha_{q_n+1}s_{q_n+1}}$, which we define as $x(s_{q_n+1})$.

Let $\sigma_1, \sigma_2 \in \mathcal{S}_5$ be any inputs such that (i) $n = n(\sigma_2) = n(\sigma_1) + 1$, (ii) for any $i(\in[1, n-1])$, $q_i = q_i(\sigma_1) = q_i(\sigma_2)$, (iii) $q_n = q_n(\sigma_2)$, and (iv) $s_{q_{n-1}+1}(\sigma_1) = B$ and $s_{q_n+1}(\sigma_2) = B$. Then, since $x(s_{q_n+1})$ is monotone (increasing or decreasing) as s_{q_n+1} increases, $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \le \max\{\frac{V_{OPT}(\sigma_1)}{V_{PQ}(\sigma_1)}, \frac{V_{OPT}(\sigma_2)}{V_{PQ}(\sigma_2)}\}$. Therefore, let $\hat{\sigma}$ be the input such that $\hat{\sigma} \in \arg\max\{\frac{V_{OPT}(\sigma_1)}{V_{PQ}(\sigma_1)}, \frac{V_{OPT}(\sigma_2)}{V_{PQ}(\sigma_2)}\}$, which means that the statement is true. □

Lemma 3.11. *The competitive ratio of PQ is at least $2 - \min_{x\in[1,m-1]}\{\frac{\alpha_{x+1}}{\sum_{j=1}^{x+1}\alpha_j}\}$.*

Proof. Consider the following input σ. Define $m' \in \arg\min_{x\in[1,m-1]}\{\frac{\alpha_{x+1}}{\sum_{j=1}^{x+1}\alpha_j}\}$. Initially, $(m'+1)B$ arrival events happen such that B packets arrive at $Q^{(1)}$ to $Q^{(m'+1)}$. Then, for $k = 1, 2, \ldots, m'$, the kth round consists of B scheduling events followed by B arrival events in which all the B packets arrive at $Q^{(m'-k+1)}$.

For σ, *PQ* transmits B packets from $Q^{(m'-k+2)}$ at the kth round. As a result, *PQ* cannot accept arriving packets in the same round. Hence, $V_{PQ}(\sigma) = B\sum_{j=1}^{m'+1}\alpha_j$ holds. On the other hand, *OPT* transmits B packets from $Q^{(m'-k+1)}$ at the kth round, and hence can accept all the arriving packets. Thus, $V_{OPT}(\sigma) = 2B\sum_{j=1}^{m'}\alpha_j + B\alpha_{m'+1}$. Therefore, $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} = \frac{2\sum_{j=1}^{m'}\alpha_j + \alpha_{m'+1}}{\sum_{j=1}^{m'+1}\alpha_j} = 2 - \frac{\alpha_{m'+1}}{\sum_{j=1}^{m'+1}\alpha_j}$. (It is easy to see that $\sigma \in \mathcal{S}_5$.) □

4. Lower bound for deterministic algorithms

In this section, we show a lower bound for any deterministic algorithm. We make an assumption that is well-known to have no effect on the analysis of the competitive ratio. We consider only online algorithms that transmit a packet at a scheduling event whenever their buffers are not empty. (Such algorithms are called *work-conserving*. See e.g. [9].)

Theorem 4.1. *No deterministic online algorithm can achieve a competitive ratio smaller than $1 + \frac{\alpha^3+\alpha^2+\alpha}{\alpha^4+4\alpha^3+3\alpha^2+4\alpha+1}$.*

Proof. Fix an online algorithm *ON*. Our adversary constructs the following input σ. Let $\sigma(t)$ denote the prefix of the input σ up to time *t*. *OPT* can accept and transmit all arriving packets in this input. $2B$ arrival events occur during time $(0, 1)$, and B packets arrive at $Q^{(1)}$ and $Q^{(m)}$, respectively. In addition, B scheduling events occur during time $(1, 2)$. For $\sigma(2)$, suppose that *ON* transmits $B(1-x)$ packets and Bx ones from $Q^{(1)}$ and $Q^{(m)}$, respectively. (See Fig. 2.) After time 2, our adversary selects one queue from $Q^{(1)}$ and $Q^{(m)}$, and makes some packets arrive at the queue.

Case 1: If $\alpha x \ge 1 - x$: B arrival events occur during time $(2, 3)$, and B packets arrive at $Q^{(1)}$. Then, the total value of packets which *ON* accepts by time 3 is $(\alpha + 1 + 1 - x)B$. Moreover, B scheduling events occur during time $(3, 4)$. For $\sigma(4)$, suppose that *ON* transmits $B(1-y)$ packets and By packets from $Q^{(1)}$ and $Q^{(m)}$, respectively. (See Fig. 3.) After time 4, in the same way as time 2, our adversary selects one queue from $Q^{(1)}$ and $Q^{(m)}$, and makes some packets arrive at the queue.

Case 1.1: If $\alpha(x + y) \ge 1 - y$: B arrival events occur during time $(4, 5)$, and B packets arrive at $Q^{(1)}$. Furthermore, $2B$ scheduling events occur during time $(5, 6)$.

For this input, $V_{ON}(\sigma) = (\alpha + 1 + 1 - x + 1 - y)B$, and $V_{OPT}(\sigma) = (\alpha + 1 + 1 + 1)B$.

Case 1.2: If $\alpha(x + y) < 1 - y$: B arrival events occur during time $(4, 5)$, and B packets arrive at $Q^{(m)}$. Moreover, $2B$ scheduling events occur during time $(5, 6)$.

For this input, $V_{ON}(\sigma) = (\alpha + 1 + 1 - x + \alpha(x + y))B$, and $V_{OPT}(\sigma) = (\alpha + 1 + 1 + \alpha)B$.

Case 2: If $\alpha x < 1 - x$: B arrival events occur during time $(2, 3)$, and B packets arrive at $Q^{(m)}$. Then, the total value of packets which *ON* accepts by time 3 is $(\alpha + 1 + \alpha x)B$. Moreover, B scheduling events occur during time $(3, 4)$. For $\sigma(4)$, *ON* transmits $B(1-z)$ packets and Bz ones from $Q^{(1)}$ and $Q^{(m)}$, respectively during time $(3, 4)$. (See Fig. 4.) After time 4, in the same way as the above case, our adversary selects one queue from $Q^{(1)}$ and $Q^{(m)}$, and causes some packets to arrive at the queue.

Case 2.1: If $\alpha z \ge 1 - x + 1 - z$: B arrival events occur during time $(4, 5)$, and B packets arrive at $Q^{(1)}$. Also, $2B$ scheduling events occur during time $(5, 6)$.

Please cite this article as: J. Kawahara et al., Tight analysis of priority queuing for egress traffic, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.09.001

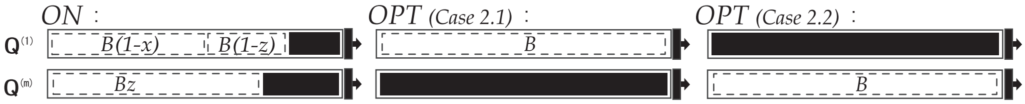

Fig. 4. States of queues at time 4 via Case 2.

For this input, $V_{ON}(\sigma) = (\alpha + 1 + \alpha x + 1 - x + 1 - z)B$, and $V_{OPT}(\sigma) = (\alpha + 1 + \alpha + 1)B$.

Case 2.2: If $\alpha z < 1 - x + 1 - z$: B arrival events occur during time (4, 5), and B packets arrive at $Q^{(m)}$. In addition, $2B$ scheduling events occur during time (5, 6).

For this input, $V_{ON}(\sigma) = (\alpha + 1 + \alpha x + \alpha z)B$, and $V_{OPT}(\sigma) = (\alpha + 1 + \alpha + \alpha)B$.

By the above argument, we define $c_1(x) = \min_y \max\{\frac{\alpha+1+1+1}{\alpha+1+1-x+1-y}, \frac{\alpha+1+1+\alpha}{\alpha+1+1-x+\alpha(x+y)}\}$ and $c_2(x) = \min_z \max\{\frac{\alpha+1+\alpha+1}{\alpha+1+\alpha x+1-x+1-z}, \frac{\alpha+1+\alpha+\alpha}{\alpha+1+\alpha x+\alpha z}\}$. Then, $\frac{V_{OPT}(\sigma)}{V_{ON}(\sigma)} \geq \min_x \max\{c_1(x), c_2(x)\}$.

$c_1(x)$ is minimized when $\frac{\alpha+1+1+1}{\alpha+1+1-x+1-y} = \frac{\alpha+1+1+\alpha}{\alpha+1+1-x+\alpha(x+y)}$. Then, $y = \frac{\alpha(\alpha+3)+(-\alpha^2-4\alpha+1)x}{\alpha^2+5\alpha+2}$. Thus, $c_1(x) \geq \frac{\alpha^2+5\alpha+2}{\alpha^2+4\alpha+2-x}$.

$c_2(x)$ is minimized when $\frac{\alpha+1+\alpha+1}{\alpha+1+\alpha x+1-x+1-z} = \frac{\alpha+1+\alpha+\alpha}{\alpha+1+\alpha x+\alpha z}$. Then, $z = \frac{\alpha^2+6\alpha+1+(\alpha^2-4\alpha-1)x}{2\alpha^2+5\alpha+1}$. Hence, $c_2(x) \geq \frac{2\alpha^2+5\alpha+1}{\alpha^2+4\alpha+1+\alpha^2 x}$.

Finally, $\min_x \max\{c_1(x), c_2(x)\}$ is minimized when $c_1(x) = c_2(x)$, that is $\frac{\alpha^2+5\alpha+2}{\alpha^2+4\alpha+2-x} = \frac{2\alpha^2+5\alpha+1}{\alpha^2+4\alpha+1+\alpha^2 x}$. Therefore, since $x = \frac{\alpha^4+4\alpha^3+2\alpha^2+\alpha}{\alpha^4+5\alpha^3+4\alpha^2+5\alpha+1}$, $\min_x \max\{c_1(x), c_2(x)\} \geq \frac{\alpha^4+5\alpha^3+4\alpha^2+5\alpha+1}{\alpha^4+4\alpha^3+3\alpha^2+4\alpha+1} = 1 + \frac{\alpha^3+\alpha^2+\alpha}{\alpha^4+4\alpha^3+3\alpha^2+4\alpha+1}$. □

5. Concluding remarks

A lot of packets used by multimedia applications arrive in a QoS switch at a burst, and managing queues to store outgoing packets (egress traffic) can become a bottleneck. In this paper, we have formulated the problem of controlling egress traffic, and analyzed Priority Queuing policies (*PQ*) using competitive analysis. We have shown that the competitive ratio of *PQ* is exactly $2 - \min_{x \in [1,m-1]}\{\frac{\alpha_{x+1}}{\sum_{j=1}^{x+1} \alpha_j}\}$. Moreover, we have shown that there is no $1 + \frac{\alpha^3+\alpha^2+\alpha}{\alpha^4+4\alpha^3+3\alpha^2+4\alpha+1}$-competitive deterministic algorithm.

We present some open questions as follows: (i) What is the competitive ratio of other practical policies, such as *WRR*? (ii) We consider the case where the size of each packet is one, namely fixed. In the setting where packets with variable sizes arrive, what is the competitive ratio of *PQ* or other policies? (iii) We are interested in comparing our results with experimental results using measured data in QoS switches. (iv) The goal was to maximize the sum of the values of the transmitted packets in this paper, which is generally used for the online buffer management problems. However, this may not be able to evaluate the actual performance of practical scheduling algorithms correctly. (We showed that the worst scenario for *PQ* is extreme in this paper.) What if another objective function (e.g., fairness) is used for evaluating the performance of a scheduling algorithm? (v) An obvious open question is to close the gap between the competitive ratio of *PQ* and our lower bound for any deterministic algorithm.

Acknowledgments

This work was supported by JSPS KAKENHI Grant number 26730008.

Appendix A. Comparing both upper counds

Our upper bound is

$$2 - \min_{x \in [1,m-1]} \left\{ \frac{\alpha_{x+1}}{\sum_{j=1}^{x+1} \alpha_j} \right\} = 1 + \max_{x \in [1,m-1]} \left\{ \frac{\sum_{j=1}^{x} \alpha_j}{\sum_{j=1}^{x+1} \alpha_j} \right\}$$

and the upper bound by Al-Bawani and Souza [2] is

$$2 - \min_{j \in [1,m-1]} \left\{ \frac{\alpha_{j+1} - \alpha_j}{\alpha_{j+1}} \right\} = 1 + \max_{j \in [1,m-1]} \left\{ \frac{\alpha_j}{\alpha_{j+1}} \right\}.$$

Now we show that

$$\max_{x \in [1,m-1]} \left\{ \frac{\sum_{j=1}^{x} \alpha_j}{\sum_{j=1}^{x+1} \alpha_j} \right\} < \max_{j \in [1,m-1]} \left\{ \frac{\alpha_j}{\alpha_{j+1}} \right\}.$$

Define $a \in \arg\max_{j \in [1,m-1]}\{\frac{\alpha_j}{\alpha_{j+1}}\}$ and $b \in \arg\max_{x \in [1,m-1]}\{\frac{\sum_{j=1}^{x} \alpha_j}{\sum_{j=1}^{x+1} \alpha_j}\}$. Then, we have that

$$\frac{\alpha_a}{\alpha_{a+1}} \geq \frac{\sum_{j=1}^{b} \alpha_j}{\sum_{j=1}^{b} \alpha_{j+1}} > \frac{\sum_{j=1}^{b} \alpha_j}{\alpha_1 + \sum_{j=1}^{b} \alpha_{j+1}} = \frac{\sum_{j=1}^{b} \alpha_j}{\sum_{j=1}^{b+1} \alpha_j}.$$

Appendix B. Restriction of input

Lemma B.1. *Let σ be an input such that OPT rejects at least one packet at an arrival event. Then, there exists an input σ' such that $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \leq \frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')}$ and OPT accepts all arriving packets.*

Proof. Let e be the first arrival event where *OPT* rejects a packet, let p be the arriving packet at e, and let t be the event time when e happens. We construct a new input σ'' by removing e from a given input σ. Then, *PQ* for σ'' might accept a packet q which is not accepted for σ after t. Suppose that *PQ* handles priorities to packets in its buffers, and transmits the packet with the highest priority at each scheduling event. Let $Q^{(i)}$ be a queue at which p arrives at e. Then, at a scheduling event after t, a priority which *PQ* handles to a packet in $Q^{(j)}$ $(j \leq i)$ for σ'' is higher than that for σ. However, a priority which *PQ* handles to a packet in $Q^{(j)}$ $(j > i)$ for σ'' is equal to that for σ. Thus, a time when a packet is transmitted from $Q^{(j)}$ $(j > i)$ in σ'' is the same as that in σ. Also, the number of packets which *PQ* stores in $Q^{(j)}$ $(j > i)$ in σ'' is equivalent to that in σ. Let k be the integer such that α_k is the value of q. Then, $i \geq k$ holds. Hence, $V_{PQ}(\sigma'') \leq V_{PQ}(\sigma)$. On the other hand, $V_{OPT}(\sigma'') = V_{OPT}(\sigma)$. According to the inequality and the equality, $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \leq \frac{V_{OPT}(\sigma'')}{V_{PQ}(\sigma'')}$. As a result, we construct

Please cite this article as: J. Kawahara et al., Tight analysis of priority queuing for egress traffic, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.09.001

a new input σ' by removing all arrival events at which *OPT* rejects a packet from σ. Then, $\frac{V_{OPT}(\sigma)}{V_{PQ}(\sigma)} \leq \frac{V_{OPT}(\sigma')}{V_{PQ}(\sigma')}$. □

References

[1] W. Aiello, Y. Mansour, S. Rajagopolan, A. Rosén, Competitive queue policies for differentiated services, J. Algorithms 55 (2) (2005) 113–141.

[2] K. Al-Bawani, A. Souza, Buffer overflow management with class segregation, Inf. Process. Lett. 113 (4) (2013) 145–150.

[3] S. Albers, T. Jacobs, An experimental study of new and known online packet buffering algorithms, Algorithmica 57 (4) (2010) 725–746.

[4] S. Albers, M. Schmidt, On the performance of greedy algorithms in packet buffering, SIAM J. Comput. 35 (2) (2005) 278–304.

[5] N. Andelman, Randomized queue management for DiffServ, in: Proceedings of the 17th ACM Symposium on Parallel Algorithms and Architectures, 2005, pp. 1–10.

[6] N. Andelman, Y. Mansour, Competitive management of non-preemptive queues with multiple values, Distrib. Comput. (2003) 166–180.

[7] N. Andelman, Y. Mansour, A. Zhu, Competitive queueing policies for QoS switches, in: Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 761–770.

[8] Y. Azar, A. Litichevskey, Maximizing throughput in multi-queue switches, Algorithmica 45 (1) (2006) 69–90.

[9] Y. Azar, Y. Richter, Management of multi-queue switches in QoS networks, Algorithmica 43 (1-2) (2005) 81–96.

[10] Y. Azar, Y. Richter, An improved algorithm for CIOQ switches, ACM Trans. Algorithms 2 (2) (2006) 282–295.

[11] Y. Azar, Y. Richter, The zero-one principle for switching networks, in: Proceedings of the 36th ACM Symposium on Theory of Computing, 2004, pp. 64–71.

[12] A. Bar-Noy, A. Freund, S. Landa, J. Naor, Competitive on-line switching policies, Algorithmica 36 (3) (2003) 225–247.

[13] M. Bienkowski, A. Madry, Geometric aspects of online packet buffering: an optimal randomized algorithm for two buffers, in: Proceedings of the 8th Latin American Theoretical Informatics, 2008, pp. 252–263.

[14] M. Bienkowski, An optimal lower bound for buffer management in multi-queue switches, Algorithmica 68 (2) (2014) 426–447.

[15] S. Blanke, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An architecture for differentiated services, in: Proceedings of the RFC2475, IETF, December 1998.

[16] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, 1998.

[17] Cisco Systems, Inc, "Campus QoS Design", http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoSDesign.html, 2014.

[18] Cisco Systems, Inc, "Cisco Catalyst 2955 series switches data sheets", http://www.cisco.com/en/US/products/hw/switches/ps628/products_data_sheets_list.html, 2014.

[19] Cisco Systems, Inc, "Cisco Catalyst 6500 series switches data sheets", http://www.cisco.com/en/US/products/hw/switches/ps708/products_data_sheets_list.html, 2014.

[20] A. Demers, S. Keshav, S. Shenker, Analysis and simulation of a fair queueing algorithm, J.. Internetw. Res. Exp. 1 (1) (1990) 3–26.

[21] M. Englert, M. Westermann, Lower and upper bounds on FIFO buffer management in QoS switches, Algorithmica 53 (4) (2009) 523–548.

[22] R. Fleischer, H. Koga, Balanced scheduling toward loss-free packet queuing and delay fairness, Algorithmica 38 (2) (2004) 363–376.

[23] M. Goldwasser, A survey of buffer management policies for packet switches, ACM SIGACT News 41 (1) (2010) 100–128.

[24] E. Hahne, A. Kesselman, Y. Mansour, Competitive buffer management for shared-memory switches, in: Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures, 2001, pp. 53–58.

[25] M. Katevenis, S. Sidiropoulos, C. Courcoubetis, Weighted round-robin cell multiplexing in a general-purpose ATM switch chip, IEEE J. Select. Area Commun. 9 (8) (October 1991) 1265–1279.

[26] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, M. Sviridenko, Buffer overflow management in QoS switches, SIAM J. Comput. 33 (3) (2004) 563–583.

[27] A. Kesselman, Y. Mansour, Harmonic buffer management policy for shared memory switches, Theor. Comput. Sci. 324 (2-3) (2004) 161–182.

[28] A. Kesselman, Y. Mansour, R. van Stee, Improved competitive guarantees for QoS buffering, Algorithmica 43 (1-2) (2005) 63–80.

[29] A. Kesselman, A. Rosén, Scheduling policies for CIOQ switches, J. Algorithms 60 (1) (2006) 60–83.

[30] A. Kesselman, A. Rosén, Controlling CIOQ switches with priority queuing and in multistage interconnection networks, J. Interconnect. Netw. 9 (1/2) (2008) 53–72.

[31] A. Kesselman, K. Kogan, M. Segal, Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing, Distrib. Comput. 23 (3) (2010) 163–175.

[32] A. Kesselman, K. Kogan, M. Segal, Best effort and priority queuing policies for buffered crossbar switches, Chicago J. Theor. Sci. (2012) 1–14.

[33] A. Kesselman, K. Kogan, M. Segal, Improved competitive performance bounds for CIOQ switches, Algorithmica 63 (1-2) (2012b) 411–424.

[34] K. Kobayashi, S. Miyazaki, Y. Okabe, A tight bound on online buffer management for two-port shared-memory switches, in: Proceedings of the 19th ACM Symposium on Parallel Algorithms and Architectures, 2007, pp. 358–364.

[35] K. Kobayashi, S. Miyazaki, Y. Okabe, A tight upper bound on online buffer management for multi-queue switches with bicodal buffers, IEICE TRANSACTIONS on Fundam. Electron., Commun. Comput. Sci. E91-D (12) (2008) 2757–2769.

[36] K. Kobayashi, S. Miyazaki, Y. Okabe, Competitive buffer management for multi-queue switches in QoS networks using packet buffering algorithms, in: Proceedings of the 21st ACM Symposium on Parallel Algorithms and Architectures, 2009, pp. 328–336.

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

[38] K. Kogan, A. Lopez-Ortiz, S.I. Nikolenko, A.V. Sirotkin, Online Scheduling FIFO policies with admission and Push-Out, Theory Comput. Syst. (2015).

[39] S.I. Nikolenko, K. Kogan, Single and multiple buffer processing, in: Encyclopedia of Algorithms, Springer, 2015, pp. 1–9.

[40] D. Sleator, R. Tarjan, Amortized efficiency of list update and paging rules, Commun. ACM 28 (2) (1985) 202–208.

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

Jun Kawahara was born in Osaka, Japan, in 1981. He received the B.S. degree in science and the M.E. and Ph.D. degrees in informatics from Kyoto University, Japan, in 2004, 2006, and 2009, respectively. In 2009-2010, he worked at Kyoto University as a Researcher. He joined the JST ERATO Minato Discrete Structure Manipulation System Project as a Researcher in 2010–2012. Currently, he is an Assistant Professor at Nara Institute of Science and Technology, Nara, Japan. His research interests include theoretical computer science and discrete algorithms.

Koji M. Kobayashi has received his B.E., M.I. and Ph.D. from Kyoto University. He is a research associate at National Institute of Informatics, Japan. His research interests include algorithms and complexity theory.

Tomotaka Maeda is a research associate in Academic Center for Computing and Media Studies, Kyoto University. He received his B.E. and M.I. degrees from Kyoto University in 2006 and 2008, respectively. His research interest is Internet architecture.

Please cite this article as: J. Kawahara et al., Tight analysis of priority queuing for egress traffic, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.09.001