# Buffer Management Algorithms in Unbounded Buffers

Jae-Hoon Kim, *Member, KIMICS*

***Abstract*— In a network router, packet loss may occur when it overflows due to sudden burst traffic. This paper studies how much large buffers are required to eliminate the packet losses. There are m buffers on which packets arrive and one output port to which a packet is transmitted at a time. The buffer management algorithm should determine one of the buffers whose packet is transmitted to the output port at each time. The front packet belonging to the buffer determined by the algorithm is transmitted. The goal is to minimize the sum of the lengths of buffers to transmit all the packets.**

**We provide an optimal off-line algorithm and also we show the lower bounds of on-line algorithms. The on-line algorithm has no prior information of the packets having arrived in the future. Its performance is compared to that of the optimal off-line algorithm.**

***Index Terms*— Buffer management, off-line algorithms, on-line algorithms, lower bound.**

## I. INTRODUCTION

In the current network, the packet losses occur when the buffers in the router are overflowed by burst traffic. As the one of the ways to prevent the packet losses, sufficiently large buffers in the routers are required. In this paper, we will study the amount of buffers in a router to eliminate the packet losses.

In a router, there are m buffers on each of which the packets arrive. At a time, one of the packets which are located in the front of the buffers is transmitted through the one output port. The buffer management algorithm determines the packet of which buffer is transmitted. The performance of the algorithm is measured by the sum of the lengths of buffers to transmit all the packets. This measure is different from that given in the previous work [1], which is the maximum length of the buffers.

In this paper, we will provide an optimal off-line algorithm for this new measure. Also we consider on-line situation, where the algorithm has no information of the packets before their arrivals. The performance of on-line algorithm is compared to that of the optimal off-line algorithm. We provide the lower bounds of the on-line algorithms Longest Queue First(LQF) and Round Robin(RR).

Manuscript received October 8, 2010; revised November 1, 2010; accepted November 10, 2010.

Jae-Hoon Kim is with the Division of Computer Engineering, Pusan University of Foreign Studies, Busan, 604-168, Korea (Email: jhoon@pufs.ac.kr)

## II. RELATED WORK

Over the past decade, there is great interest in the study of buffer management algorithms in the context of packet transmission on network switches. As a typical model, a switch receives packets on one or more input ports, where each packet has a designated output port through which it should be transmitted, see survey papers [2], [3], [4], [5].

The model considered in this paper was given in [1], where the buffers have no limit of their lengths. But the authors in [1] are concerned in minimizing the maximum length of buffers. They show that the on-line algorithm LQF is $\Theta(\log m)$-competitive.

For the case that the lengths of buffers are bounded by some constant B, there are many previous works. First, we can consider the case where there is one buffer and one output port. In this case, each packet has a value and the goal is to maximize the sum of values of the packets transmitted to the output port. The nonpreemptive model in which the packets accepted in the buffer should be transmitted is studied in [6], [7]. Also there are the works [8], [9] for the preemptive model, where the packets accepted in the buffer can be dropped later.

There are the other models [10], [11], where there are either multiple buffers and one output port or multiple input ports and multiple output ports. In [10], a model similar to our work is studied in which there are multiple buffers and one output port but the lengths of buffers are bounded. In [11], there is given a model where there are multiple input ports and multiple output ports, called CIOQ switch.

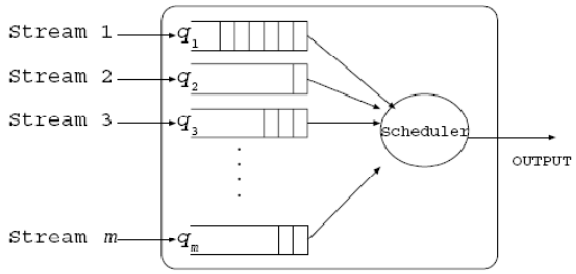


Fig 1. The router model.

## III. MODEL

In Figure 1, there are m buffers on which packets destined to one output port are stored temporarily. At each time, an arbitrary number of packets arrive on each buffer, and a buffer management algorithm determines one buffer among those buffers, the front packet of which is transmitted to the output port.

We will divide a time into two phases. One is *the arrival phase* in which the packets arrive on each buffer. The other is *the transmission phase* when one of the front packets of the buffers is transmitted. The algorithm should determine the packet of which buffer is transmitted before the second phase.

The buffers are assumed to have unbounded lengths, that is, the size of the buffers may be extended in amount of our needs. Therefore all packets arriving on buffers may be stored without rejections.

## IV. OPTIMAL OFF-LINE ALGORITHM

In this section, we provide an off-line algorithm $\mathcal{A}$ and we prove that it is optimal. Here we will describe the algorithm $\mathcal{A}$.

Let $h_i(t)$ be the height of buffer $i$ after the transmission phase of time $t$, that is, the number of packets remained in buffer $i$ at time $t$. Also $\ell_i(t)$ denote the length of buffer $i$ after the transmission phase of time $t$, that is, it satisfies that there is a $1 \le t_0 \le t$ such that $\ell_i(t) = h_i(t_0)$ and $\ell_i(t) \ge h_i(t')$ for all $t'$, $1 \le t' \le t$.

We divide the buffers into two parts, called *live* buffers and *dead* buffers, at time $t$. Let $a_i(t)$ be the total number of all the packets which will arrive in the buffer $i$ after time $t$. A buffer $i$ is called to be live if $h_i(t) + a_i(t) > \ell_i(t)$. Otherwise, that is, $h_i(\mathrm{t}) + a_i(t) \le \ell_i(t)$, it is called to be dead. In other words, for a live buffer, if there is no transmission on the buffer after time $t$, then its length will be increased, while for a dead buffer, its length remains unchanged. So after time $t$ when a buffer is dead, a transmission on the buffer is worthless.

Let $last_i$ be the last time when a packet arrives on the buffer $i$. The algorithm $\mathcal{A}$ selects a buffer on which a packet is transmitted among the buffers in an increasing order of their $last_i$'s. So, for convenience, we arrange the buffers in an increasing order of their $last_i$'s. First, we start with the buffer 1, that is, the buffer with the smallest $last_i$. The algorithm $\mathcal{A}$ transmits the packet on buffer 1 until it is dead. Of course, if there is empty in buffer 1, then select the nonempty buffer with the smallest index after 1. If $\mathcal{A}$ transmits the packet on a buffer $i$ at time $t$ and the buffer $i$ becomes dead at that time, then $\mathcal{A}$ will transmit the packet on the buffer $i+1$ from the next time of $t$. Therefore in $\mathcal{A}$, if a buffer becomes dead at time $t$, then there is no transmission on it after $t$. Also $\mathcal{A}$ transmits a packet on a live buffer at each time. Consequently, at each time, $\mathcal{A}$ transmits a packet on the buffer with the lowest index among the live buffers.

In the following theorem, we will prove that $\mathcal{A}$ is optimal. That is, $\mathcal{A}$ minimizes the sum of lengths of buffers for all the possible instances of packets.

**Theorem 1** The algorithm $\mathcal{A}$ given in the above is an optimal off-line algorithm.

**Proof:** Let $OPT$ be an optimal off-line algorithm. Consider the time $t$ when $\mathcal{A}$ and $OPT$ first choose different buffers, $i$ and $k$, respectively. Then we can consider an algorithm $OPT'$ which behaves in the same way of $OPT$ and selects $i$ instead of $k$ only at time $t$.

To show that $OPT'$ has a same performance as $OPT$, there are two cases:

Case: $k < i$.

If buffer $k$ is empty after the arrival phase of time $t$, then $OPT'$ may be better than $OPT$. It is a contradiction. Otherwise, by the definition of time $t$, $\mathcal{A}$ has the same length of buffer $k$ as $OPT$ after the arrival phase of time $t$ and before the transmission phase of time $t$. For the case of $\mathcal{A}$, it will transmit no packet in the buffer $k$ at time $\ge t$, because it selects buffers in an increasing order of their indices. It says that the buffer $k$ is dead. So the lengths of the buffer $k$ in $OPT'$ and $OPT$ remains same after time $t$. But in the case of $OPT'$, there is a possibility to decrease the length of the buffer $i$ since $OPT'$ transmits a packet on the buffer $i$ at time $t$. Thus, $|OPT'| \le |OPT|$, where $|A|$ denotes the sum of lengths

of buffers in the algorithm $A$.

Case: $k > i$.

Let $\ell_i^{OPT}(\ell_i^{OPT'})$ and $\ell_k^{OPT}(\ell_k^{OPT'})$ be the last length of buffer $i$ and $k$, respectively, in $OPT(OPT'$, resp.). Since $\mathcal{A}$ transmits a packet on the buffer $i$ at time $t$, both the buffer $i$ and $k$ are live. Thus, $\ell_i^{OPT'} = \ell_i^{OPT} - 1$ and $\ell_k^{OPT'} = \ell_k^{OPT} + 1$. So, $\ell_i^{OPT'} + \ell_k^{OPT'} = \ell_i^{OPT} + \ell_k^{OPT}$. Consequently, $|OPT'| = |OPT|$.

If we repeat the above argument, then we can show that $\mathcal{A}$ has a same performance as $OPT$.

## V. ON-LINE ALGORITHMS

In this section, we will provide the lower bounds of on-line algorithms for the buffer management problem, where the on-line algorithms have no information of packets arriving in the future in advance.

Here we consider two on-line algorithms such as Longest Queue First(LQF) and Round Robin(RR). In LQF, at each time $t$, the algorithm transmits the packet on the buffer with the highest height at $t$. If there are two more buffers with the highest height, then choose one of them arbitrarily. Also in RR, the algorithm considers the buffers in the non-decreasing order of their indices. At each time, it selects the buffer with the next smallest index which is not empty. That is, if the algorithm selected the queue $q_i$ at time t, then the queue selected at time $t+1$ is $q_{(r+i) \bmod m}$, where $r$ is the smallest positive integer satisfying that $q_{(r+i) \bmod m}$ is not empty.

We will prove that both LQF and RR are $\Omega(\log m)$-competitive. In the next theorem, we will consider the instances to derive the lower bound.

**Theorem 2** The on-line algorithms LQF and RR are both $\Omega(\log m)$-competitive.

**Proof:** First, we consider the instance $\mathcal{J}$ to derive the lower bound of LQF. In $\mathcal{J}$, at time 0, $(2h+1)$ packets arrive on buffer 1, where $h$ is the positive integer determined later. Next, at each time $t$ between time 1 and time $h$, one packet arrives on buffer 2. Then at each time from 0 to $h$, LQF transmits the packet on buffer 1. After the transmission phase of time $h$, there are $h$ packets both on buffer 1 and 2 in LQF. Next, at time $t$ between time $(h+1)$ and time $2h$, one packet arrives alternately on buffer 3 and buffer 4. Then LQF transmits the packets only on buffer 1 and buffer 2. Thus, after the transmission phase of time $2h$, $h/2$ packets remain on each buffer from 1 to 4. In general, at time $t$ between $kh+1$ and $(k+1)h$ $(k \geq 1)$, one packet arrives alternately on the buffer from $(2^k+1)$ to $2^{k+1}$. Then LQF transmits the packets only on buffer from 1 to $2^k$. Thus, after the transmission phase of time $(k+1)h$, $h/2^k$ packets remain on each buffer from 1 to $2^{k+1}$.

Choose the integer $i$ such that $\frac{h}{2^i} = 1$, that is, $i = \log h$. Then after the transmission phase of time $(\log h + 1)h$, only one packet remains on each buffer from 0 to $2^{(\log h+1)}$. After this time, no packet arrives. Also we can choose the integer $h$ to satisfy that $2^{(\log h+1)} = m$, that is, $h = m/2$.

Therefore, we consider the performance of LQF on $\mathcal{J}$. For $\mathcal{J}$, the lengths of buffer 1 and 2 are equal to $(2h+1)$ and $h$, respectively, in LQF. For $1 \leq k \leq i$, the lengths of the buffers from $(2^k+1)$ to $2^{k+1}$ are equal to $h/2^k$. Thus,

$$\begin{aligned}\|\mathrm{LQF}\| &= 2h + 1 + h + \sum_{k=1}^{i} 2^k \cdot \frac{h}{2^k} \\ &= 3h + 1 + ih = (3+i)h+1\end{aligned}$$

We consider the performance of the off-line optimal algorithm $OPT$ on $\mathcal{J}$. At each time t between 1 and $(i+1)h$, $OPT$ transmits the packet having arrived at $t$. Thus the lengths of buffers from 1 to $m$ are equal to 1. So,

$$\|OPT\| = 2h + 1 + m = 4h + 1.$$

Therefore, we get the lower bound as follows:

$$\frac{\|\mathrm{LQF}\|}{\|OPT\|} \geq \frac{(3+i)h+1}{4h+1} \geq \frac{i}{4} = \frac{\log h}{4} = \frac{1}{4}\log\frac{m}{2}.$$

Next, we consider the on-line algorithm RR. The instance $\mathcal{J}'$ different from $\mathcal{J}$ only at time 0 is given in which $h+1$ packets arrive at time 0 in buffer 1. Then we can see that after the transmission phase of time $kh$, $h/2^k$ packets remain on each buffer from 1 to $2^k$, where $1 \leq k \leq i$. Thus we can choose an integer $h$ such that $2^{\log h} = m$. Also, for $\mathcal{J}'$, the lengths of buffer 1 and 2 are equal to $h+1$ and $h/2$, respectively, in RR. For $1 \leq k \leq i-1$, the lengths of the buffers from $(2^k+1)$ to $2^{k+1}$ are equal to $h/2^{k+1}$. So,

$$\|\mathrm{RR}\| = h + 1 + h + \sum_{k=1}^{i-1} 2^k \cdot \frac{h}{2^{k+1}} = (3+i)\frac{h}{2} + 1.$$

But, $\|OPT\| = h + 1 + (m-1) = h + m = 2h$. Thus,

$$\frac{\|\mathrm{RR}\|}{\|OPT\|} \geq \frac{1}{4}\log h = \frac{1}{4}\log\frac{m}{2}.$$

## VI. CONCLUSIONS

In this paper, we provide the optimal off-line algorithm and the lower bounds of on-line algorithms for the buffer management problem. We are concerned in minimizing the sum of lengths of buffers, which is different from the previous work [1]

As the further work, the analysis of the upper bounds of the on-line algorithms is needed.

## REFERENCES

[1] Rudolf Fleischer and Hisashi Koga, "Balanced Scheduling toward Loss-Free Packet Queuing and Delay Fairness", *Algorithmica*, vol. 38(2), pp. 363-376, 2003.
[2] Yossi Azar, "Online packet switching", *Proc. Second Workshop on Approximation and Online Algorithms*, pp. 1-5, 2004.
[3] Leah Epstein and Rob van Stee, "Buffer management problems", *SIGACT news*, vol. 35(3), pp. 58-66, 2004.
[4] Wojciech Jawor, "Three dozen papers on online algorithms", *SIGACT news*, vol. 36(1), pp. 71-85, 2005.
[5] Marek Chrobak, "Online algorithms column 13", *SIGACT news*, vol. 39(3), pp. 96-121, 2008.
[6] William A. Aiello, Yishay Mansour, S. Rajagopolan, and Adi Rosen, "Competitive queue policies for differentiated services", *Journal of Algorithms*, vol. 55(2), pp. 113-141, 2005.
[7] An Zhu, "Analysis of queuing policies in QoS switches", *Journal of Algorithms*, vol. 53(2), pp. 137-168, 2004.
[8] Yishay Mansour, Boaz Patt-Shamir, and Ofer Lapid, "Optimal smoothing schedules for real-time streams", *Distributed Computing*, vol. 17(1), pp. 77-89, 2004.
[9] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko, "Buffer overflow management in QoS switches", *SIAM Journal on Computing*, vol. 33(3), pp. 563-583, 2004.
[10] Yossi Azar and Yossi Ritcher, "Management of multi-queue switches in QoS networks", *Proc.35th ACM Symp. on Theory of Computing*, pp. 82-89, 2003.
[11] Alex Kesselman and Adi Rosen, "Scheduling policies for CIOQ switches", *Journal of Algorithms*, vol. 60(1), pp. 60-83, 2006.

**Jae-Hoon Kim**
Received his B.S. degree in Mathematics from Sogang University in 1994, his M.S. degree in Mathematics from KAIST in 1996, and his Ph.D. degree in Computer Science from KAIST in 2003. He is currently an associate professor at Division of Computer Engineering, Pusan University of Foreign Studies. His research interests include on-line algorithms, scheduling, and computational geometry