アルゴリズム 72－1
（2000．3．21）

遅延を考慮したジッタレギュレータの解析

古賀 久志
富士通研究所

〒211-8588 川崎市中原区上小田中 4-1-1
E-mail: koga@flab.fujitsu.co.jp

マルチメディアデータをネットワーク経由で配送して円滑に再生するには個々のパケットの遅延のゆらぎ（ジッタ）を抑えることが大切である。本稿では、配送経路途中にあるルータにおいて、パケットをバッファリングすることによりジッタを吸収するオンラインアルゴリズムを取り扱う。

従来研究ではルータ内のバッファ数のみに着目してアルゴリズムを解析していた。本稿ではそれに加えて、通信のリアルタイム性を考慮し、ルータ内にパケットを保持できる時間に上限があるという条件を新規に導入して解析を行った。その結果、オンラインアルゴリズムの competitiveness はバッファ数よりもむしろルータ内のパケット存在時間に依存するという結果を得た。また、我々が提案したオンラインアルゴリズムがジッタをどれだけ定量的に吸収するかについても考察を与えた。

An Analysis for Jitter Regulators with Delay Consideration

Hisashi Koga
Fujitsu Laboratories Ltd.

4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki 211-8588, Japan
E-mail: koga@flab.fujitsu.co.jp

In order to playback multimedia data smoothly, it is important to keep the jitter, the variability of delay of individual packets, low. This paper examines on-line algorithms in a single router on the distribution path for a given multimedia stream which attempt to regulate jitter by holding packets in an internal buffer.

The previous work solved the problem with focusing only on the buffer size in the router. However, for the purpose of providing the stream communication with real-time property, this paper introduces the new condition that a packet can stay in the router at most for a constant time which we name *the permitted delay time* into the problem besides the conventional constraint about the buffer size. Our analysis of this new version of the problem obtains the result that the competitiveness of on-line algorithms depends on the permitted delay time rather than the buffer size. Finally we investigate quantitatively how much jitter is removed by our on-line algorithm.

1 Introduction

With the rapid expansion of network communication to public represented by the Internet, there is a growing need for networks with guaranteed quality of service (QoS). Whereas the traditional best-effort networks like the current Internet does not give guarantees on communication performance to the clients, QoS networks allow them to transfer information with a certain level of performance guarantees which are evaluated in terms of delay, delay jitter, throughput and packet loss rate.

Delay jitter which we refer to simply as jitter hereafter is defined as the maximal gap between delay of each packet in the given stream. Keeping jitter low is important for many real-time applications which continuously transmit multimedia data for the reason below. Ideally, in a real-time application, it is desirable that packets which are generated at the source in a periodic fashion should arrive to the destination also periodically, which means the jitter is equal to 0. However, in reality, as packets pass through intermediate switches or routers, the jitter increases due to variable queuing delays at these network equipments and variable propagation delays over network links. As the result, for stable playback of multimedia data, the real-time applications must prepare a buffer in the destination host to absorb the jitter. The large jitter forces the destination host to prepare a huge buffer. The jitter should be kept low also from the viewpoint of the network management.

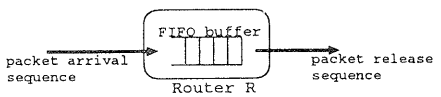

Figure 1: Jitter regulation at a router

This paper focuses on the jitter regulation at a single router for a given stream in the context that, by holding the incoming packets in an internal buffer, the router attempts to output the packets at the same intervals whose lengths are X (Figure 1). Here, X denotes the length of the interval between adjacent packets at the source generating packets periodically. As packets go through the network before reaching the router, the packet intervals become distant from X. We may consider that the router tries to reconstruct the original traffic pattern at the source. The router is assumed to know the value of X *a priori*. Mansour and Patt-Shamir [3] deals with on-line algorithms for this problem under the condition that the number of spaces constituting the internal buffer is restricted, where each space can exactly store one packet. Packets in the buffer must be pushed out obeying a FIFO service discipline, when the buffer becomes full. They presented an on-line algorithm using $2B$ buffer spaces which obtains the same jitter as the optimal off-line algorithm using B spaces. As for lower bounds, they proved that, in case on-line algorithms have the same buffer capacity B as the optimal off-line algorithm, there exist arrival sequences for which the off-line algorithm achieves 0-jitter and any on-line algorithm gets a jitter at least BX.

However, they examine the problem allowing a packet to stay in the router infinitely long unless the buffer is not full. In fact, their proof for the lower bounds utilizes the technique that an off-line algorithm postpones arbitrarily long sending out a packet, after the on-line algorithm outputs the very packet. We argue here that this assumption is not realistic to satisfy the requirement of some types of real-time applications. For example, keeping a packet long in the routers on the way between the source and the destination increases the end-to-end delay very much. This disables the bi-directional interactive communication demanded by real-time discussion services, *etc.*

Hence, to get closer to practical use, this paper introduces the new condition that a packet can stay in the router at most for a constant time L to the jitter regulation problem besides the conventional condition about the buffer capacity. That is, if L time units have passed since a packet arrival, the corresponding packet is thrown out of the router. In a normal situation, $L \gg X$. Specifically we name this constant L as the *permitted delay time*. This new problem is named *jitter regulation problem with delay consideration (JRDC)* hereafter.

This paper examines the power of on-line algorithms for JRDC. We analyze and evaluate on-line algorithms with competitive analysis [4] which compares the performance of an on-line algorithm with that of the optimal off-line algorithm. First in Section 3 we introduce the optimal off-line algorithm which is derived by simply extending the optimal off-line algorithm by Mansour and Patt-Shamir [3] for the case without delay consideration. In Section 4 the lower bounds of the competitiveness of on-line algorithms are investigated. We show, for any on-line algorithm, a packet arrival sequence exists such that the jitter incurred by the on-line algorithm is larger than or equal to that of the optimal off-line algorithm plus $\frac{L}{2} - 2X$. In Section 5, we present a simple on-line algorithm with the nearly tight upper bound, i.e. its jitter

does not exceed that of the optimal off-line algorithm by $\frac{L}{2}$. Interestingly, the competitiveness for JRDC depends only on the permitted delay-time and not on the number of buffer spaces. We also make clear quantitatively how much jitter our online algorithm absorbs against individual packet arrival sequences. This result is valuable in practice, because the competitive analysis itself does not give actual quantitative bounds, thus not being utilized in applied research fields. We show that, provided the buffer size is sufficient, our algorithm decreases the jitter L at its best. Especially if the jitter of the packet arrival sequence is less than $\frac{L}{2}$, the output stream gets 0-jitter.

We conclude the introduction by contrasting this paper with practical QoS processing proposed in the applied research fields on communication. In the QoS system proposed in the applied research fields, routers with jitter control function consist of two components: regulators and a scheduler [1] [2], [5]. The regulators shape the input traffic into the desired traffic pattern by holding packets in a buffer, where one regulator is responsible for one input stream. When a packet is released from a regulator, it is passed to the scheduler which decides the order of transmitting packets from all the streams to the output link. This system has two primary features.

1. The routers on the distribution path for a given stream cooperate such that a regulator of a router holds packets long enough to absorb the jitter generated at the scheduler of the previous router whose size grows up to the maximal scheduling delay in the worst case.
2. The amount of the total traffic is restricted by the admission control to suppress the scheduling delay. When there is a request to establish a new stream connection, the entire network judges whether it can afford to accommodate the new stream.

However these two features are difficult to realize in the networks like the Internet managed by a lot of different organizations. It is probable that only a part of routers support the QoS mechanism, so the adjacent routers cannot work together. Furthermore it seems also hard to limit the total traffic flowing into the scheduler so as to bound the maximal scheduling delay.

This paper studies regulators. We concentrate on a single router without considering the entire network. This approach is reasonable for this reason that the admission control over the entire network like the current Internet seems impossible.

2 Problem Statement

The jitter-regulation problem is formally defined as follows. We are given a router R and a sequence of packets $p_1, p_2, \ldots p_n$ which arrive to R. Each p_k $(1 \leq k \leq n)$ arrives at time $a(k)$. Upon their arrivals, R stores the packets into an internal buffer and releases them later. A jitter-regulation algorithm A decides the release time of packets. The release time of p_k by the algorithm A is denoted as $s_A(k)$. R must release packets obeying the FIFO service discipline. That is, $s_A(k) \leq s_A(k+1)$ for $1 \leq k \leq n-1$. Throughout this paper we refer to the sequence $a(1), a(2), \ldots, a(n)$ as the *arrival time sequence* and the sequence $s_A(1), s_A(2), \ldots s_A(n)$ as the *release time sequence* by the algorithm A.

The purpose of a jitter-regulation algorithm is to make the release time sequence near to the ideal time sequence in which all the packets are spaced X time units apart. R knows the value of X before the transmission of the stream starts. The jitter of a time sequence σ: $t_1, t_2, \ldots, t_n$ is defined as follows.

$$J^\sigma = \max_{0 \leq i,k \leq n} \{|t_i - t_k - (i-k)X|\}. \qquad (1)$$

That is, this variable measures the distance between σ and the ideal time sequence. Remark that we can calculate the jitter not only for the release time sequence but also for the arrival time sequence. We express the jitter of the release time sequence by the algorithm A for the arrival time sequence σ as $J_A(\sigma)$. Mansour and Patt-Shamir [3] mentions the next property concerning the jitter without proof.

Lemma 1 ([3]) *For all m $(1 \leq m \leq n)$,*

$$\begin{aligned} J^\sigma &= \max_{0 \leq i \leq n} \{t_i - t_m - (i-m)X\} \\ &\quad + \max_{0 \leq i \leq n} \{t_m - t_i - (m-i)X\} \end{aligned}$$

Proof: First we prove that the value of J^σ is not influenced, even if the absolute value brackets are removed from the definition (1). Suppose that i, k are a pair of indices which maximizes J^σ and that $t_i - t_k - (i-k)X \leq 0$. Then, by swapping i and k, we find another pair of indices which achieves the same value of J^σ and satisfies $t_i - t_k - (i-k)X \geq 0$. Hence $J^\sigma = \max_{0 \leq i,k \leq n}\{t_i - t_k - (i-k)X\}$. It follows that

$$\begin{aligned} J^\sigma &= \max_{0 \leq i,k \leq n} \{t_i - t_k - (i-k)X\} \\ &= \max_{0 \leq i \leq n} \{t_i - iX\} + \max_{0 \leq k \leq n} \{kX - t_k\} \end{aligned}$$

$$
\begin{aligned}
&= \max_{0\le i\le n}\{t_i - iX\} - (t_m - mX) \\
&\quad + \max_{0\le k\le n}\{kX - t_k\} + (t_m - mX) \\
&= \max_{0\le i\le n}\{t_i - t_m - (i-m)X\} \\
&\quad + \max_{0\le i\le n}\{t_m - t_i - (m-i)X\}.
\end{aligned}
$$

□

This paper examines the jitter regulation problem under the next two conditions one of which is newly introduced in this paper.

1. The number of spaces in an internal buffer is exactly B.
2. The permitted delay time is equal to L for every packet.

The first condition says that R can store at most B packets simultaneously. Before p_{k+B} arrives, p_k needs to be released. The second condition means that each packet cannot stay at R more than L time units. Normally L is far greater than X. In summary, we solve the problem under the condition for the release time that

$$a(k) \le s_A(k) \le \min\{a(k+B), a(k)+L\}, \quad (2)$$

where we define $a(k) = \infty$ for $k > n$.

Because Mansour and Patt-Shamir [3] does not consider the permitted delay time, they solve the problem under the release time condition that

$$a(k) \le s_A(k) \le a(k+B). \quad (3)$$

Instead, if the buffer size is large enough to assure that the buffer never becomes full, it is natural to examine the problem under the next condition, focusing only on the permitted delay time,

$$a(k) \le s_A(k) \le a(k) + L. \quad (4)$$

In this way, in terms of the interests in algorithmic theory, setting a different condition for the release time yields a different problem.

3 Optimal Off-line Algorithm

Our on-line algorithms are compared with the optimal off-line algorithm OFF in the subsequent sections. We present OFF here. OFF is derived by simply extending the optimal off-line algorithm for the case without delay consideration [3].

Algorithm OFF:

1. Let E_k be the time interval such that $E_k = [a(k) - (k-1)X, \min\{a(k+B), a(k)+L\} - (k-1)X]$, for $1 \le k \le n$.
2. OFF finds a minimal interval M which intersects all the intervals E_k for $1 \le k \le n$. Define $P_k = \min(E_k \cap M)$.
3. OFF releases the k-th packet p_k at time $S_{OFF}(k) = P_k + (k-1)X$.

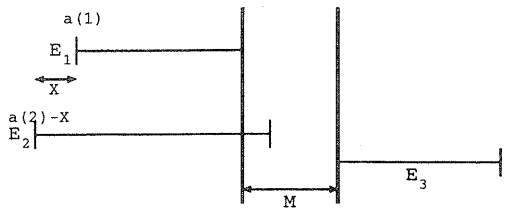

Figure 2: An example of E_k

Figure 2 gives an example of E_k for three consecutive packets p_1, p_2, p_3, where p_1 and p_2 arrive simultaneously and p_3 arrives R extremely later than the first two packets. Since p_1 and p_2 arrive at the same time, $\min(E_2) = \min(E_1) + X$. The region M is partitioned off by two vertical thick lines.

Theorem 1 *OFF is an off-line algorithms which achieves the minimal jitter.*

Proof: First we prove that OFF is a valid algorithm by showing OFF satisfies the release time condition (2) and the FIFOness. Since $s_{OFF}(k) = \min(E_k \cap M) + (k-1)X$, it is obvious that $s_{OFF}(k) \in [a(k), \min\{a(k+B), a(k)+L\}]$. Thus the release time condition is (2) satisfied. With respect to the FIFOness, we need to show

$$P_{k+1} + X \ge P_k, \quad (5)$$

as $s_{OFF}(k) = P_k + kX$ and $s_{OFF}(k+1) = P_{k+1} + (k+1)X$. We need to consider the following two cases.
(Case I) $\min(M) \ge \min(E_k)$: In this case, P_k is equal to $\min(M)$ which is less than $\min(E_{k+1} \cap M) = P_{k+1}$. Thus (5) is proved.
(Case II) $\min(M) < \min(E_k)$: In this case, $P_k = \min(E_k)$. Therefore,

$$
\begin{aligned}
P_{k+1} + X &= \min(E_{k+1} \cap M) + X \\
&\ge \min(E_{k+1}) + X \\
&= a(k+1) - (k-1)X \\
&\ge a(k) - (k-1)X = \min(E_k) = P_k
\end{aligned}
$$

Thus, (5) is verified. This concludes the proof for the FIFOness. From the minimality of M, OFF achieves the minimal jitter. □

4 The Lower Bound

In this section, the lower bound on the competitiveness of on-line algorithms is derived as Theorem 2. In the proof, an adversary constructs a arrival time sequence σ which annoys on-line algorithms, observing the action by the on-line algorithms.

Theorem 2 *For any on-line algorithm A, there exists a arrival time sequence σ for which $J_{OFF}(\sigma) = 0$, while $J_A(\sigma)$ becomes at least $\frac{L}{2} - 2X - \epsilon$ when $B \geq 2\lfloor\frac{L}{X}\rfloor + 1$, where ϵ is a constant which can be arbitrarily close to 0.*

The lower bound is proved for particular values of B. However, since, in the next section, we present an on-line algorithm whose jitter is larger at most by $\frac{L}{2}$ than OFF without regard to the value of B, this theorem is sufficient to insist that our on-line algorithm is nearly optimal. As a matter of of fact, the values of B are chosen so that the buffer never becomes full when OFF and A process σ.

Proof: The request sequence σ is constructed by an adversary in the next way. The adversary passes the first packet p_1 at time 0 to R. Then it passes the subsequent packets at the same interval of X, until A releases p_1. Let T be the time A releases p_1. From the release time condition (2), $T \leq L$. The number of packets passed to R before T is expressed as $\lfloor\frac{T}{X}\rfloor + 1$ from the construction of σ. The adversary alters how to construct the rest of σ according to whether $T > \frac{L}{2}$ or not. In a word, if $T > \frac{L}{2}$, the adversary constructs σ such that $s_{OFF}(1) = 0$. Otherwise it attempts to organize σ such that $s_{OFF}(1) = L$.

(Case I) Suppose that $T \leq \frac{L}{2}$.
The adversary passes the $(\lfloor\frac{T}{X}\rfloor + 2)$-th packet to R at time $L + (\lfloor\frac{T}{X}\rfloor + 1)X$. Since the number of packets composing σ is $\lfloor\frac{T}{X}\rfloor + 2$, the buffer overflow never takes place, because $\lfloor\frac{T}{X}\rfloor + 2 \leq 2\lfloor\frac{L}{X}\rfloor + 1 \leq B$. From the definition of jitter (1),

$$\begin{aligned} J_A(\sigma) &\geq |s_A(\lfloor\frac{T}{X}\rfloor + 2) - s_A(1) - (\lfloor\frac{T}{X}\rfloor + 1)X| \\ &\geq L + (\lfloor\frac{T}{X}\rfloor + 1)X - T - (\lfloor\frac{T}{X}\rfloor + 1)X \\ &= L - T \geq \frac{L}{2}. \end{aligned}$$

On the other hand, OFF gets 0-jitter by releasing p_k at time $L + (k-1)X$. Thus the theorem holds.

(Case II) Suppose that $T > \frac{L}{2}$.
The adversary puts in the next $\lfloor\frac{L}{X}\rfloor$ packets at time T', where T' is an arbitrary time in the interval $(T, (\lfloor\frac{T}{X}\rfloor + 1)X)$. In this case the number of packets included in σ is $\lfloor\frac{T}{X}\rfloor + \lfloor\frac{L}{X}\rfloor + 1$ which is less than $2\lfloor\frac{L}{X}\rfloor + 1$. Thus, the buffer overflow does not occur. OFF gets 0-jitter by releasing p_k at $(k-1)X$, for $1 \leq k \leq \lfloor\frac{T}{X}\rfloor + \lfloor\frac{L}{X}\rfloor + 1$. On the other hand, the jitter incurred by A, $J_A(\sigma)$ is at least

$$\begin{aligned} & |s_A(\lfloor\frac{T}{X}\rfloor + \lfloor\frac{L}{X}\rfloor + 1) - s_A(1) - (\lfloor\frac{T}{X}\rfloor + \lfloor\frac{L}{X}\rfloor)X| \\ \geq\ & |(T' + L) - T - (\lfloor\frac{T}{X}\rfloor + \lfloor\frac{L}{X}\rfloor)X| \\ \geq\ & (\frac{T}{X} + \frac{L}{X} - 2)X + T - T' - L \\ =\ & T - 2X + (T - T') > \frac{L}{2} - 2X + (T - T') \end{aligned}$$

Since $T - T'$ can be arbitrarily close to 0, the theorem holds for (Case II) by replacing $T - T'$ with ϵ. Thus, the entire proof completes. □

5 The Upper Bound

This section pursues on-line algorithms which have an upper bound on the competitiveness. First we present a technique which narrows the scope of the candidate algorithms. After that, our on-line algorithm named $HALF$ is introduced among the set of candidate algorithms. For any arrival time sequence, $HALF$ gets a jitter greater than OFF by at most $\frac{L}{2}$.

The next theorem claims that, in designing an efficient on-line algorithm, we should concentrate on when it releases p_1. The strategy to choose the release times for packets after p_1 can be decided in a rather routine fashion.

Theorem 3 *On condition that an on-line algorithm A releases p_1 at time $s_A(1)$, A gets the best jitter by choosing the release times for packets after p_1 in the following way. Let $s^*(k) = s_A(1) + (k-1)X, (1 \leq k \leq n)$.*

$$s_A(k) = \begin{cases} s^*(k) & \textit{(TYPE A)} \\ \text{if } a(k) \leq s^*(k) \leq \min\{a(k+B), a(k)+L\} & \\ a(k) & \textit{(TYPE B)} \\ \text{if } s^*(k) < a(k) & \\ \min\{a(k+B), a(k)+L\} & \textit{(TYPE C)} \\ \text{if } s^*(k) > \min\{a(k+B), a(k)+L\} & \end{cases}$$

Proof: It is obvious that A behaves as an on-line algorithm. Let σ be an arrival time sequence. From Lemma 1, $J_A(\sigma)$ is equal to

$$\max_{0\le k\le n}\{s_A(k) - s_A(1) - (k-1)X\} + \max_{0\le k\le n}\{s_A(1) - s_A(k) - (1-k)X\} \quad (6)$$

Note that both the first term and the second term in the right side of (6) cannot be negative, because they are 0 when substituting 1 for k. When an index k satisfies $s_A(1) + (k-1)X \in [a(k), \min\{a(k+B), a(k)+L)]$, both terms of the right side of (6) equal 0 by setting $s_A(k)$ to $s_A(1) + (k-1)X$, and, hence, this index k does not contribute to the increase of the jitter. This case corresponds to (TYPE A) illustrated in Theorem 3. Otherwise, when an index k satisfies $s_A(1) + (k-1)X < a(k)$, the second term is less than 0 and unrelated to increasing the jitter. To minimize the first term, $s_A(k)$ must be chosen as small as possible. Therefore $s_A(k)$ must be $a(k)$, which corresponds to (TYPE B). Finally, when k satisfies $s_A(1)+(k-1)X > \min\{a(k+B), a(k)+L\}$, the first term is less than 0 and irrelevant to increasing the jitter. To reduce the second term $s_A(k)$ must be as large as possible and chosen to $\min\{a(k)+L, a(k+B)\}$. This case is associated with (TYPE C). □

If the release time condition is replaced with (3) for the case without considering delay, the similar argument yields the next corollary immediately. For this version of the jitter regulation problem, no on-line algorithm which present the upper bound on the competitiveness has been found so far unfortunately, when on-line algorithms have the same amount of buffer as the off-line algorithm. However this corollary may shed light on this difficulty.

Corollary 1 *On condition that an on-line algorithm A which serves the case without considering delay releases packet at time $s_A(1)$, A gets the best jitter by choosing the release times for packets after p_1 in the following way. Let $s^*(k) = s_A(1) + (k-1)X, (1 \le k \le n)$.*

$$s_A(k) = \begin{cases} s^*(k), & \text{if } a(k) \le s^*(k) \le a(k+B) \\ a(k), & \text{if } s^*(k) < a(k) \\ a(k+B), & \text{if } s^*(k) > a(k+B) \end{cases}$$

Among the algorithms fulfilling the behavior specification in Theorem 3, we present our on-line algorithm named $HALF$ which gets the jitter larger than OFF by at most $\frac{L}{2}$.

Algorithm $HALF$: This algorithm release p_1 at time $\min\{a(1+B), a(1)+\frac{L}{2}\}$. After that, it constructs the release time sequence obeying the rule mentioned in Theorem 3, where $s^*(k) = \min\{a(1+B), a(1)+\frac{L}{2}\} + (k-1)X$.

Theorem 4 *For any arrival time sequence σ,*

$$J_{HALF}(\sigma) <= J_{OFF}(\sigma) + \frac{L}{2}. \quad (7)$$

Theorem 2 and Theorem 4 together claim the competitiveness of on-line algorithms for JRDC is dominated by the permitted delay time L and is not affected by the buffer size B, despite we incorporate both B and L into the problem statement. In other words, these theorems clarify the difference between JRDC and the case without considering delay [3] where its lower bound becomes BX when both on-line algorithms and the optimal off-line algorithm have the same buffer size B.

Proof: The proof proceeds as divided into two cases depending on whether $J_{OFF}(\sigma) = 0$. Recall the notation in the proof of Theorem 1. To save the space, we only write H instead of the algorithm name $HALF$, when it appears in mathematical expressions.

(Case I) $J_{OFF}(\sigma) \neq 0$: In this case the width of the region M is not 0. Suppose i is the value of the index k which maximizes $\max\{a(k)-(k-1)X\}$ and j is the one which minimizes $\min\{a(k+B), a(k)+L\} - (k-1)X\}$. Since the width of M is not 0, we have $a(i) - (i-1)X > \min\{a(j+B), a(j)+L\} - (j-1)X$. $J_{OFF}(\sigma)$ is expressed as:

$$a(i) - (i-j)X - \min\{a(j+B), a(j)+L\} \quad (8)$$

As for $HALF$, we classify the values of k into three sets S_A, S_B, S_C by the behavior of $HALF$.

$$\begin{aligned} S_A &= \{k : s_H(k) = s_H(1) + (k-1)X\} \\ S_B &= \{k : s_H(k) > s_H(1) + (k-1)X\} \\ S_C &= \{k : s_H(k) < s_H(1) + (k-1)X\} \end{aligned}$$

From the description of $HALF$, S_A, S_B and S_C correspond to (TYPE A), (TYPE B) and (TYPE C) respectively. As is stated in the proof of Theorem 3, regarding to the right side of the formula (6), we have

- The indices included in S_A make both of the first and the second terms get the value 0.
- The first term is made positive only by the indices in S_B.
- On the contrary, the second term is made positive only by the indices in S_C.

Therefore (6) may be transformed into the next expression.

$$\begin{aligned} J_H(\sigma) &= \max\{0, \max_{k\in S_B}\{s_H(k) - s_H(1) - (k-1)X\}\} \\ &+ \max\{0, \max_{k\in S_C}\{s_H(1) - s_H(k) + (k-1)X\}\}, \end{aligned}$$

where we define $\max_\phi\{s_H(k)-s_H(1)-(k-1)X\} = \max_\phi\{s_H(1) - s_H(k) + (k-1)X\} = -1$. From now on, we start to show $J_H(\sigma) \leq J_{OFF}(\sigma) + \frac{L}{2}$. There are three cases depending on the relation between $\min\{a(1)+\frac{L}{2}, a(1+B)\}$ and the open space $(\min\{a(j+B), a(j)+L\}-(j-1)X, a(i)-(i-1)X)$.

- If $\min\{a(1+B), a(1)+\frac{L}{2}\} \in (\min\{a(j+B), a(j)+L\}-(j-1)X, a(i)-(i-1)X)$, we have $i \in S_B$, and $j \in S_C$. Thus neither S_B nor S_C is an empty set. In this case, $J_H(\sigma)$ is computed as follows.

$$\begin{aligned} &\max_{k\in S_B}\{s_H(k) - s_H(1) - (k-1)X\} \\ &+ \max_{k\in S_C}\{s_H(1) - s_H(k) + (k-1)X\} \\ &= \max_{k\in S_B}\{a(k) - (k-1)X\} \\ &\quad + \max_{k\in S_C}\{(k-1)X - \min\{a(k+B), a(k)+L\}\} \\ &= a(i) - (i-j)X - \min\{a(j+B), a(j)+L\} \end{aligned}$$

This indicates $J_{OFF}(\sigma) = J_H(\sigma)$.

- If $\min\{a(1+B), a(1)+\frac{L}{2}\} > a(i) - (i-1)X$, we have $S_B = \phi$ because for any k, $a(k) - (\min\{a(1+B), a(1)+\frac{L}{2}\}+(k-1)X) \leq a(i) - (i-1)X - \min\{a(1)+\frac{L}{2}, a(1+B)\} < 0$. We can calculate $J_H(\sigma)$ in the next way. Note $a(i) - (i-1)X \geq a(1) - (1-1)X = a(1)$.

$$\begin{aligned} &\max_{k\in S_C}\{s_H(1) - s_H(k) + (k-1)X\} \\ &= s_H(1) + (j-1)X - \min\{a(j+B), a(j)+L\} \\ &\leq a(1) + \frac{L}{2} + (j-1)X - \min\{a(j+B), a(j)+L\} \\ &\leq a(i) - (i-j)X - \min\{a(j+B), a(j)+L\} + \frac{L}{2} \\ &= J_{OFF}(\sigma) + \frac{L}{2}. \end{aligned}$$

The proof for this case is completed.

- If $\min\{a(1+B), a(1)+\frac{L}{2}\} < \min\{a(j+B), a(j)+L\} - (j-1)X$, we have $S_C = \phi$ because for any k, $\min\{a(1+B), a(1)+\frac{L}{2}\} + (k-1)X - \min\{a(k)+L, a(k+B)\} \leq \min\{a(1+B), a(1)+\frac{L}{2}\}+(j-1)X-\min\{a(j)+L, a(j+B)\} < 0$. In addition, we have $s_H(1) = a(1) + \frac{L}{2}$. This is because, if we assume $a(1)+\frac{L}{2} > a(1+B)$, the right edge of E_1 becomes $a(1+B) < \min\{a(j+B), a(j)+L\}-(j-1)X$, which contradicts with the premise that M intersects with E_1. Thus $J_H(\sigma) =$

$$\begin{aligned} &\max_{k\in S_B}\{a(k) - (k-1)X - s_H(1)\} \\ &\leq a(i) - (i-1)X - (a(1) + \frac{L}{2}) \\ &\leq a(i) - (i-1)X - \min\{a(1+B), a(1)+L) + \frac{L}{2} \\ &\leq a(i) - (i-j)X - \min\{a(j+B), a(j)+L\} + \frac{L}{2} \\ &= J_{OFF}(\sigma) + \frac{L}{2} \end{aligned}$$

(Case II) $J_{OFF}(\sigma) = 0$: Our purpose for this case is to show $J_H(\sigma) = \frac{L}{2}$. Since the width of M is 0, we have that $a(k)-(k-1)X < \min\{a(1+B), a(1)+L\}$ and that $a(1) < \min\{a(k+B), a(k)+L\}-(k-1)X$ for any k. First we show that either $S_B = \phi$ or $S_C = \phi$. Suppose that $S_B \neq \phi$ and $S_C \neq \phi$. Then there exists a pair of indices p, q such that:

1. $a(p) > s_H(1) + (p-1)X$.
2. $\min\{a(q+B), a(q)+L\} < s_H(1) + (q-1)X$.

Therefore $\min\{a(q+B), a(q)+L\} - (q-1)X < a(p) - (p-1)X$. However this inequality indicates the width of M is not 0, contradicting with the fact $J_{OFF}(\sigma) = 0$. Hence at least one of S_B and S_C must be empty. There are two cases whether $S_B = \phi$ or $S_C = \phi$. We mention only for the case $S_C = \phi$ here due to space limitation. If $S_C = \phi$, $J_H(\sigma)$ is bounded from the above as follows.

$$\begin{aligned} J_H(\sigma) &= \max_{k\in S_B}\{s_H(k) - s_H(1) - (k-1)X\} \\ &\leq \min\{a(1+B), a(1)+L\} \\ &\quad - \min\{a(1+B), a(1)+\frac{L}{2}\} \leq \frac{L}{2} \end{aligned}$$

Since we have proved (7) regardless of the value of $J_{OFF}(\sigma)$, the whole proof finishes. □

Although we have evaluated $HALF$ by the competitiveness, these results are not necessary sufficient for router designers, since the competitive analysis does not produce actual quantitative bounds. The actual router designer will have more interests in how much jitter is removed by $HALF$ from arrival time sequences. We answer this question by insisting that $HALF$ decreases the jitter

by L at its best from an arrival time sequence, in case the size of the buffer is sufficient. Hereafter we denote the release time sequence for the arrival time sequence σ by $HALF$ as σ_H.

Theorem 5 *Pick up an arbitrary arrival time sequence σ. If the buffer size is sufficiently large that any buffer overflow does not occur, $J^{\sigma_H} - J^{\sigma} = L$ at the best of $HALF$. Especially, if $J^{\sigma} < \frac{L}{2}$, $J^{\sigma_H} = 0$.*

Proof: Since B is sufficiently large, we may replace $\min\{a(k+B), a(k)+L\}$ simply with $a(k)+L$ in the description of $HALF$. For the same reason, $HALF$ release p_1 at time $a(1)+\frac{L}{2}$. For $1 \leq k \leq n$, we define two sequences $b(k) = a(k) - (k-1)X$ and $c(k) = s_H(k) - (k-1)X$. Let i (and j) be the index which minimizes (respectively, maximizes) $b(k)$. Then we have

$$J^{\sigma} = b(j) - b(i). \tag{9}$$

As for the release time sequence, first we show that the index i minimizes $c(k)$ also. Since $a(i) - (i-1)X \leq a(1) \leq s_H(1)$, $i \in S_A$ or $i \in S_C$. If $i \in S_A$, we have $a(k)+L-(k-1)X \geq a(i)+L-(i-1)X \geq s_H(1)$ for any k. Hence $S_C = \phi$ and i becomes the index which minimizes $c(k)$. If $i \in S_C$, we have $a(i)+L-(i-1)X \leq a(k)+L-(k-1)X$ for any k. Therefore i is the index minimizing $c(k)$ among the indices in S_C, which in turn implies i minimizes $c(k)$ among all indices. In the same way, we can prove easily that j is the index which maximizes $c(k)$ and that either $j \in S_A$ or $j \in S_B$. Therefore J^{σ_H} is expressed as:

$$J^{\sigma_H} = c(j) - c(i). \tag{10}$$

Next we obtain the concrete value of $c(i)$ and $c(j)$. About $c(i)$, if $i \in S_A$, $c(i) = a(1) + \frac{L}{2} = b(1) + \frac{L}{2}$. Otherwise $c(i) = b(i) + L$. Hence $c(i) = \min\{b(1) + \frac{L}{2}, b(i) + L\}$. About $c(j)$, if $j \in S_A$, $c(j) = a(1) + \frac{L}{2} = b(1) + \frac{L}{2}$. Otherwise $c(j) = b(j)$. Hence, $c(j) = \max\{b(j), b(1) + \frac{L}{2}\}$. By substituting these formulas for $c(i)$ and $c(j)$ in (10), the amount of the jitter absorbed by $HALF$, $J^{\sigma} - J^{\sigma_H}$, is calculated in the next way.

$$\begin{aligned} & b(j) - b(i) \\ - \; & (\max\{b(j), b(1) + \frac{L}{2}\} - \min\{b(i) + L, b(1) + \frac{L}{2}\}) \\ = \; & \min\{L, b(1) + \frac{L}{2} - b(i)\} + \min\{0, b(j) - b(1) - \frac{L}{2}\} \\ = \; & \min\{\frac{L}{2}, b(1) - b(i)\} + \min\{\frac{L}{2}, b(j) - b(1)\}. \end{aligned}$$

We conclude that the jitter is decreased by up to $\frac{L}{2}$ in both directions with $b(1)$ being centered. So the jitter is absorbed by L at its best. Especially if $b(j) - b(i) < \frac{L}{2}$, we have $b(1) - b(i) < \frac{L}{2}$ and $b(j) - b(1) < \frac{L}{2}$. Thus the amount of the jitter absorbed by $HALF$ is equal to $b(j) - b(i)$ and the jitter is completely removed for this case. □.

6 Concluding Remarks

In this paper we examined the jitter regulation problem on the realistic condition that a packet must be released from the router within the permitted delay time L in addition to the conventional restriction on the buffer size, which is named *the JRDC problem*. We presented a on-line algorithm for $JRDC$ whose jitter is larger than OFF by at most $\frac{L}{2}$. Furthermore we proved that the competitiveness of on-line algorithms for $JRDC$ depends only on L and not on the buffer size by obtaining the comparable lower bound. One natural open problem is to make clear how much jitter $HALF$ can absorb while allowing buffer overflows to take place from time to time. Another interesting open problem is to control jitter for multiple streams while handling the interrelation between them in the scheduler.

References

[1] N. R. Figueira and J. Pasquale. Leave-in-time: A new service discipline for real-time communications in a packet-switching network. In *Proceedings of ACM SIGCOMM'95*, pages 207–218, 1995.

[2] S. J. Golestani. A stop-and-go queueing framework for congestion management. In *Proceedings of ACM SIGCOMM'90*, pages 8–18, 1990.

[3] Y. Mansour and B. Patt-Shamir. Jitter control in QoS networks. In *Proceedings of 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 50–59, 1998.

[4] R.E. Tarjan and D.D. Sleator. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28:202–208, 1985.

[5] H. Zhang and D. Ferrari. Rate-controlled static-priority queuing. In *Proceedings of IEEE INFOCOM'93*, pages 227–236, 1993.