International Journal of Electrical and Electronic Engineering & Telecommunications

ISSN 2319 – 2518 www.ijeetc.com
Vol. 2, No. 2, April 2013
© 2013 IJEETC. All Rights Reserved

Research Paper

A NEW ALGORITHM FOR JITTER CONTROL IN WIRELESS NETWORKS FOR QUALITY OF SERVICE

B Santosh Kumar[1]*, D Sreekanth[2] and G Rajitha[2]

*Corresponding Author: **B Santosh Kumar,** ✉ santoshpbk@gmail.com

In wireless networks Jitter is a quality of service parameter that has to be reduced at switches. The jitter is classified into two: Delay jitter and Rate Jitter. The Delay jitter bounds the maximum difference in the total delay of different packets whereas the Rate jitter bounds the difference in packet delivery rates at various times. The paper analyses two algorithms namely off-line and on-line algorithms and proposes a new algorithm for reducing the rate jitter at low buffer sizes. The new algorithm, tolerance based jitter control algorithm has been compared to on-line algorithm and proved that it gives a better performance than the on-line algorithm presented in.

Keywords: Rate jitter, Buffer size, Inter departure time, Process delay

INTRODUCTION

Jitter measures the variability of delay of packets in the given stream, which is an important property for many applications (for example, streaming real-time applications). Ideally, packets should be delivered in a perfectly periodic fashion; however, even if the source generates an evenly spaced stream, unavoidable jitter is introduced by the network due to the variable queuing and propagation delays, and packets arrive at the destination with a wide range of inter-arrival times. The jitter increases at switches along the path of a connection due to many factors, such as conflicts with other packets wishing to use the same links, and non-deterministic propagation delay in the data-link layer. Jitter is quantified in two ways. One measure, called delay jitter, bounds the maximum difference in the total delay of different packets (assuming, without loss of generality, that the abstract source is perfectly periodic). This approach is useful in contexts such as interactive communication (e.g., voice and video tele-conferencing), where guarantees on the delay jitter can be translated to the maximum buffer size needed

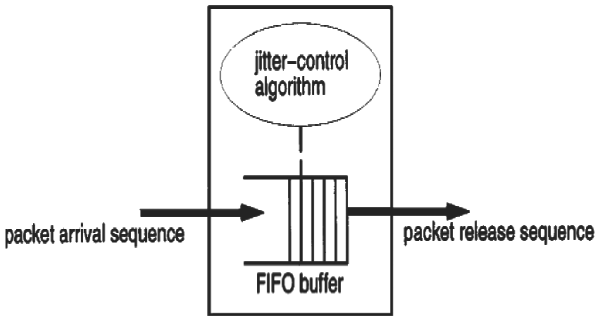
[1] Department of EIE, GITAM University, Hyderabad, Andhra Pradesh, India.
[2] Department of ECE, LIET, Hyderabad, Andhra Pradesh, India.

at the destination. The second measure, called rate jitter, bounds the difference in packet delivery rates at various times. More precisely, rate jitter measures the difference between the minimal and maximal inter-arrival times (inter-arrival time between packets is the reciprocal of rate). Rate jitter is a useful measure for many real-time applications, such as a video broadcast over the net; a slight deviation of rate translates to only a small deterioration in the perceived quality. The paper analyses offline and online algorithms for controlling the rate jitter and proposes a new algorithm for reducing the jitter at low buffer sizes.

THE BUFFER MODEL

Let us consider the following abstract communication model for a node in the network as shown in Figure 1. Here, a sequence of packets denoted by 0, 1, 2, 3, …, n, is given where each packet arrives at time $a(k)$. These packets are assumed to have equal size. Each packet is stored in the buffer upon arrival, and is released some time (perhaps immediately) after its arrival. Packets are released in FIFO order. The time of packet release (also called packet departure or packet send) is governed by a jitter control algorithm. Given an algorithm A and an arrival time sequence, we denote by $S_A(k)$, the time in which packet is released by A.

Figure 1: Jitter Control Model. The Jitter-Control Algorithm Controls Packet Release from the Buffer, Based on the Arrival Sequence



Consider jitter control algorithms which use bounded-size buffer space. Each buffer slot is capable of storing exactly one packet. All packets must be delivered, and hence the buffer size limitation can be formalized as follows. The release time sequence generated by algorithm A using a buffer of size B must satisfy the following condition for all $0 \le k \le n$:

$$a(k) \le S_A(k) \le a(k + B)$$

where $a(k) = \infty$ for $k \ge n$.

The lower bound expresses the fact that a packet cannot be sent before it arrives, and the upper bound states that when packet $k + B$ arrives, packet must be released due to the FIFOness and limited size of the buffer. A sequence of departure times is said to be B—feasible for a given sequence of arrival times if it satisfies (2a), i.e., it can be attained by an algorithm using buffer space. An algorithm is called on-line if its action at time t is a function of the packet arrivals and releases which occur before or at t; an algorithm is called off-line if its action may depend on future events, too. A times sequence is a non-decreasing sequence of real numbers. The properties of time sequences are defined below.

DEFINITIONS

Given a times sequence $\sigma = \{t_i\}_{i=0}^{n}$, its average, minimum and maximum inter-arrival times can be described as follows.

- The average inter-arrival time of σ is:

$$X_A^{\sigma} = \frac{t_n - t_o}{n}$$

- The minimum inter-arrival time of σ is

$$X^{\sigma}_{\min} \min\{t_{i+1} - t_i \mid 0 \le k < n\}$$

- The maximum inter-arrival time of σ is

$$X^{\sigma}_{\max} \max\{t_{i+1} - t_i \mid 0 \le k < n\}$$

The superscript σ has been omitted when the context is clear. The average rate of is simply $1/X^{\sigma}_{A}$. Note that since the definitions are given for a single sequence, "inter-arrival times" may sometimes mean inter-departure time, depending on the context. The main concern here is about the rate jitter of σ, which can be described intuitively as the maximal difference between inter-arrival times, which is equivalent to the difference between rates at different times. Formally, the rate jitter of σ is defined to be:

$$\max_{0 \le i,\ j \le n} \left\{ \left| (t_{i+1} - t_i) - (t_{j+1} - t_j) \right| \right\}$$

The means for analyzing the performance of jitter-control algorithms is competitive analysis (Jagadish and Manivasakan, 2011).

OFF-LINE AND ONLINE RATE JITTER ALGORITHMS

Suppose a complete sequence $\{a(k)\}_{k=0}^{n}$ of packet arrival times is given. To find a sequence of release times $\{S_{off}(k)\}_{k=0}^{n}$ which minimizes the rate jitter, using no more than B buffer space. The off-line algorithm is defined as follows:

Algorithm A (Offline)

- For each $0 \le k \le n$, define the interval

  $E_k = [a(k) - kX_{avg}, a(k + B) - kX_{avg}]$

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

- Find an interval M of minimal length which intersects all intervals E_k.
- For each packet k, let $P_k = \min(E_k \cap M)$, and define

  $S_{off}(k) = P_k + kX_{avg}$

Algorithm B (Online)

The algorithm uses $B_{on} = 2B + h$ buffer space. With each possible number $0 \le j \le 2B + h$ of packets in the buffer, we associate an inter-departure time denoted by $IDT(j)$, defined as follows. Let $\delta = (I_{\max} - I_{\min})/h$

$$IDT(j) = \begin{cases} I_{\max} & \text{if } 0 \le j \le B \\ I_{\max} - (j - B)\delta & \text{if } B \le j \le B + h \\ I_{\min} & \text{if } B + h \le j \le 2B + h \end{cases}$$

Note that $IDT(j)$ is a monotonically decreasing function in j.

- The algorithm starts with a buffer loading stage, in which packets are only accumulated and not released until the first time that the number j of packets in the buffer satisfies $IDT(j) \le X_{avg}$.
- Let = min $\{j | IDT(j) \le X_{avg}\}$, and let T^* denote the first time in which the number of packets in the buffer reaches S.
- At time T^*, the loading stage is over: the first packet is released and the following rule governs the remainder of the execution of the algorithm.
- A variable last_departure is maintained, whose value is the time at which the last packet was sent.
- If at time t, $t \le$ last_departure + $IDT(j)$, where is the number of packets currently in the buffer, then a packet is delivered and last departure is updated.

SLIP-OPTIMIZED RATE JITTER CONTROL ALGORITHM

Algorithm C (SORJC)

- Load first B packets into the buffer and update the count j.
- When $j = B$ for the first time, release the first packet from the buffer and update j.$s(1) = a(B)$.
- Release the next k^{th} packet at time $s(k)$, given by

$$s(k) = \begin{cases} a(B) + (k-1)X_{avg} + \delta & \text{if } a(B+k) > a(B) + k \cdot X_{avg} \\ a(B) + (k-1)X_{avg} & \text{otherwise} \end{cases}$$

where, $\delta = (B - j)\dfrac{X_{avg}}{2B}$ and update k.

The sequentially incoming packets are received and stored in the buffer, which is of size $2B$. Firstly, these packets are not released until the time, when the number of packets stored in the buffer becomes equal to B. This is known as the buffer loading stage. After the loading stage is over and when the number of packets in the buffer reaches B, the first packet is released at time instant, $s(1) = a(B)$.

The above equation states that the first packet is sent at the same time the B^{th} packet arrives. The remainder of the algorithm follows the following condition:

$$s(k) = \begin{cases} a(B) + (k-1)X_{avg} + \delta & \text{if } a(B+k) > a(B) + k \cdot X_{avg} \\ a(B) + (k-1)X_{avg} & \text{otherwise} \end{cases}$$

where, $\delta = (B - j)\dfrac{X_{avg}}{\beta}$ j is the number of packets in the buffer at a given instant of time and β is a constant which should always be greater than $B + 1$. As shown above, δ depends on the difference between constant B, variable j and β. δ is designed in such a way that the rate jitter of the system is reduced. The value of δ varies being positive, negative or zero.

Suppose, when $a(B+k) > a(B) + k \cdot X_{avg}$

$$s(k) = a(B) + (k-1)X_{avg} + \delta$$

Similarly, $s(k+1) = a(B) + k \cdot X_{avg} + \delta$

When the number of packets in the buffer (j) becomes equal to B (i.e., $j = B$), then δ becomes zero ($\delta = 0$). This implies:

$$\begin{aligned} s(k+1) &= a(B) + k \cdot X_{avg} \\ &= a(B) + (k-1)X_{avg} + X_{avg} \\ &= s(k) + X_{avg} \end{aligned}$$

In this case, $s(k + 1)$ becomes independent of δ and depends only on X_{avg} and $s(k)$ which means that the packets are released constantly from the buffer at a rate of X_{avg}, which indirectly states that the rate jitter is zero. Thus, in order to keep the rate jitter minimum, we tend to maintain the value of δ close to zero. The δ becomes zero when the number of packets in the buffer is equal to B as proved earlier. Therefore, in order to keep δ close to zero we try to maintain the number of packets in the buffer to be close to B. If j becomes greater than B, then the packets are released at a relatively higher rate such that the number of packets in the buffer becomes equal to B packets. By doing this, we are also taking care that there should not be any overflow in the buffer. Similarly, in case j becomes less than B, then the packets are released at a relatively slow rate. By this, we try that there should not be any underflow at the buffer.

ADVANTAGES OF ALGORITHM C OVER ALGORITHM B

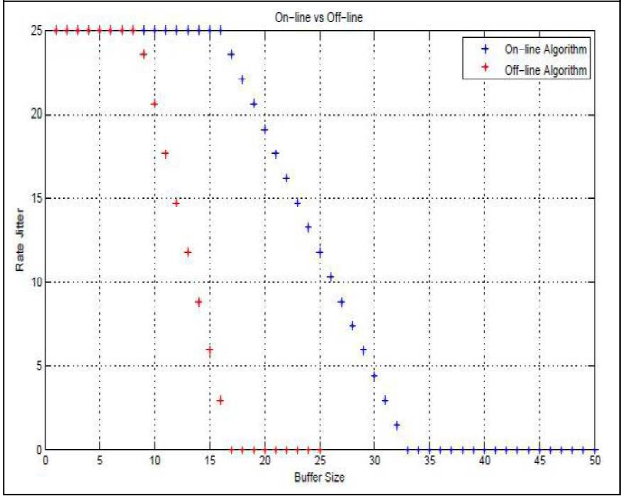
- This algorithm requires only one parameter from the arrival sequence, i.e., X_{avg}, unlike the Algorithm *B* which requires three parameters like I_{max}, I_{min} and X_{avg} from the off-line algorithm (Algorithm *A*).
- Algorithm *C* is independent of the off-line Algorithm *A*, whereas Algorithm *B* is dependent.
- Algorithm *C* is less complex and requires lesser number of computations compared to Algorithm *B*.

SIMULATION RESULTS

Off-Line vs. On-Line Algorithms

Figure 2 shows the comparison of rate jitters of off-line and on-line algorithms. It can be observed from the figure that, the on-line algorithm achieves approximately the same rate jitter as off-line algorithm with the size of the buffer equal to $2B + h$ (where B is the buffer size of off-line algorithm and h is an arbitrary constant).

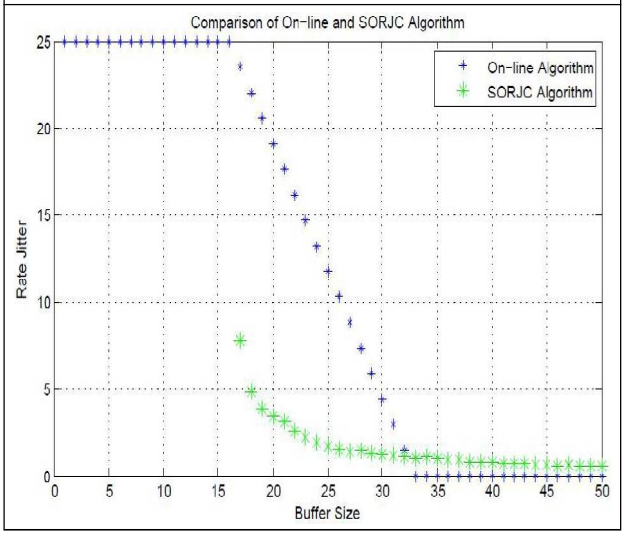
Figure 2: Off-Line vs. On-Line Algorithms



On-Line Algorithm vs. Algorithm *C*

Figure 3 shows the comparison of on-line algorithm with the new efficient Algorithm *C*. It can be observed from the figure that the rate jitter of Algorithm *C* is very less for small buffer sizes compared to on-line algorithm.

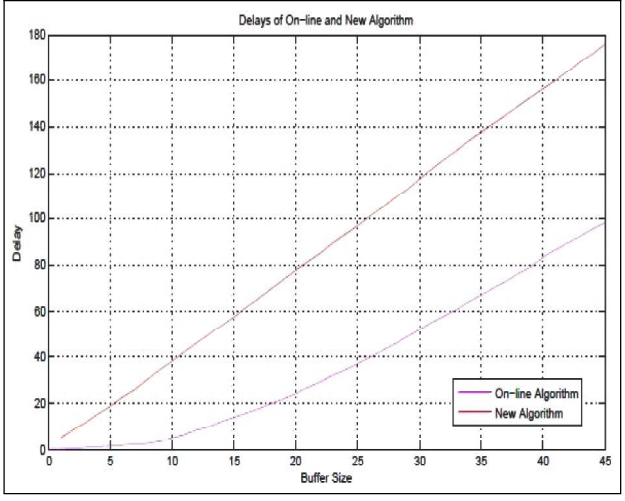
Figure 3: On-Line Algorithm vs. Algorithm *C*



Delays of On-Line Algorithm and Algorithm *C*

The rate jitter in Algorithm *C* is reduced compared to online algorithm but at the cost

Figure 4: Delay Comparison of Algorithm *C* and On-Line Algorithm



of a small increase in the overall delay of the system. It can be seen from the above Figure 4, the delay of Algorithm *C* is more than delay of On-line algorithm thus giving out a better rate jitter which is the main interest of this thesis.

CONCLUSION

The off-line and on-line algorithms have been successfully implemented and presented in this thesis. Also, the rate jitter of off-line and on-line algorithms has been compared. The on-line algorithm achieves approximately the same rate jitter as the on-line algorithm by using a buffer size of $2B + h$ (where B is the buffer size off-line algorithm). Based on the above algorithms, a new and effective algorithm has been designed and implemented considering the drawbacks of the above mentioned algorithms. The new algorithm, tolerance based jitter control algorithm has been compared to on-line algorithm and proved that it gives a better performance than the on-line algorithm presented in.

REFERENCES

1. Jagadish S and Manivasakan R (2011), “Analysis of Jitter Control Algorithm in Qos Networks”, *IEEE Transactions on Networking*, December 29.
2. Yishay Mansour and Boaz Patt-Shamir (2001), “Jitter Control in QoS Networks”, *IEEE/ACM Transactions on Networking*, Vol. 9, No. 4, pp. 492-502.