Available online at www.elixirpublishers.com (Elixir International Journal)

Electrical Engineering

Elixir Elec. Engg. 74 (2014) 26637-26645

ISSN: 2229-712X

Analysis of distributed delay jitter Control in QOS networks

K.Rameshchandra*, A K.C Varma, Ch.V.V.S.Srinivas and G.Prasannakumar
ECE/Vishnu Institute of Tech/ Jntuk/Bhimavaram, India.

ARTICLE INFO

Article history:
Received: 22 June 2013;
Received in revised form:
20 August 2014;
Accepted: 29 August 2014;

Keywords
Buffer overflow and under flow,
Jitter control,
Quality of service networks.

ABSTRACT

We study jitter control in networks with guaranteed quality of service (QoS) from the competitive analysis (as mentioned in [1]) point of view. we analyze on-line algorithms for single jitter regulator that control jitter and compare their performance to the best possible by an off-line algorithm as proposed in [1]. For delay jitter, where the goal is to minimize the difference between delay times of different packets, we show that a simple on-line algorithm using a buffer of B slots guarantees the same delay jitter as the best off-line algorithm using buffer space B/2. We prove that the guarantees made by our (proposed in [1]) on-line algorithm hold, even for simple distributed implementations, where the total buffer space is distributed along the path of the connection, provided that the input stream satisfies a certain simple property. The significance of the results is that it proves the on-line algorithm to be the best possible algorithm to reduce delay jitter for a given buffer size B. The main argue is even if both the distributed and non distributed algorithms get same jitter which one has more advantage. We focused on the advantages of distributing the buffers. The algorithm in its original form is applicable only to a fixed number of packets. We extend the results to a more practical model in which we compare off-line algorithm with n inputs and on-line algorithms with n_1 (>n) inputs.

Introduction

The need for networks with guaranteed quality of service (QoS) is widely recognized today. Unlike today's best effort networks such as the Internet, where the user has no guarantee on the performance it may expect from the network, QoS networks guarantee the end-user application a certain level of performance. For example, ATM networks support guaranteed QoS in various parameters, including end-to-end delay and delay jitter. 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 nondeterministic 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 a guarantee on the delay jitters can be translated to the maximum buffer size needed 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. In this work, we will be considering only delay jitter.

For delay jitter, they [1] have given a very simple on-line algorithm, and proved that the delay jitter in its output is no more than the delay jitter produced by an optimal (off-line) algorithm using half the space. They have given a lower bound on delay jitter, showing that doubling the space is necessary. They also consider a distributed implementation of our algorithm, where the total space of 2B is distributed along a path. They proved that the distributed algorithm guarantees the same delay jitter of a centralized off-line algorithm using space B, provided that an additional condition on the beginning of the sequence is met. To complete the picture, we also describe an efficient optimal off-line algorithm. For all our delay-jitter algorithms, they assume that the average inter-arrival time

Tele:
E-mail addresses: godiprasanna@gmail.com

of the input stream (denoted Xa) is given ahead of time. This assumption is natural for real-time connections (for example, it is included in the ATM standard). One way to view the relativistic guarantee of our algorithm is the following. Assume that the specific arrival sequence is such that using a buffer of size one can reduce the jitter completely (i.e., zero jitter). In such a case, our online algorithm, using space would also output a completely periodic sequence (i.e., zero jitter).

The main objective of the paper is to prove that the guarantees made by our (proposed in [1]) on-line algorithm hold, even for simple distributed implementations, where the total buffer space is distributed along the path of the connection, provided that the input stream satisfies a certain simple property. We did a simulation part for that proof. We get expressions for the end-to-end delay bounds (maximum and minimum delays) for the distributed off-line and on-line algorithms. The main drawback of paper [1] is that the input sequence (say of length n) considered is same for both the algorithms. Hence, the results are valid only for those n arrival sequences. In practice the receiver has packets coming in infinitely. So an attempt has been made to incorporate the same.

The remaining sections of this paper are organized as follows. In Section 2 we discussed jitter control model and give the basic definitions and notations. In section 3 we study delay jitter for a single switch. In Section 4, we study distributed delay jitter control model. In Section 5, we find an expression for the end-to-end delays encountered by packets in distributed case. In section 6, we will show the simulation results and finally in Section 7, we draw some conclusions.

Jitter control model

Consider the following abstract communication model for a node in the network as shown in the figure A sequence of packets denoted 0, 1, 2, ..., n, where each packet k arrives at time a(k) are given. 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 k is released by A.

Consider jitter control algorithms which use bounded-size buffer space. Assume that 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 $\forall\ 0 \leq k \leq n$:

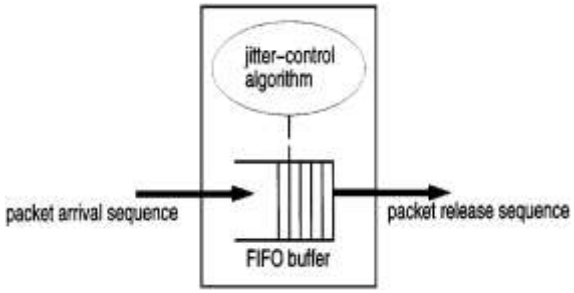

Figure 2.1: Jitter control model. The jitter-control algorithm controls packet release from the buffer, based on the arrival sequence.

$$a(k) \leq s_A(k) \leq a(k + B) \qquad (2.1)$$

Where define $a(k) = \infty$ for $k > n$. 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 k must be released due to the FIFOness and limited size of the buffer. A sequence of departure times is called B-feasible for a given sequence of arrival times if it satisfies (2.1), i.e., it can be attained by an algorithm using buffer space B. An algorithm is called on-line if its action at time is a function of the packet arrivals and releases which occur before or at; 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. Given a times sequence $\sigma = \{t_i\}_{i=0}^{n}$ its average, minimum, and maximum inter-arrival times are defined as follows.

• The average inter-arrival time of σ is

$$X_a^\sigma = \frac{t_n - t_0}{n} \quad (2.2)$$

• The minimum inter-arrival time of σ is

$$X_{min}^\sigma = \min\{t_{i+1} - t_i | 0 \leq k < n\} \quad (2.3)$$

• The maximum inter-arrival time of σ is

$$X_{min}^\sigma = \max\{t_{i+1} - t_i | 0 \leq k < n\} \quad (2.4)$$

The superscript σ has been omitted when the context is clear. The *average rate* of σ is simply $\frac{1}{X_a^\sigma}$. 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 delay jitter, intuitively, measures how far off is the difference of delivery times of different packets from the ideal time difference in a perfectly periodic sequence, where packets are spaced exactly Xa time units apart. Formally, given a times sequence $\sigma = \{t_i\}_{i=0}^n$, we define the delay jitter of to be

$$j_\sigma = \max_{0 \leq i,k \leq n}\{|t_i - t_k - (i-k)X_a|\} \quad (2.5)$$

The means used for analyzing the performance of jitter-control algorithms is competitive analysis. Here, the delay jitter of the sequence produced by an on-line algorithm against the best jitter attainable for that sequence is measured. As expected, finding the release times which minimize jitter may require knowledge of the complete arrival sequence in advance, i.e., it can be computed only by an off-line algorithm. Our results are expressed in terms of the performance of our on-line algorithms using buffer space B_{on} as compared to the best jitter attainable by an off-line algorithm using space B_{off} , where usually $B_{off} < B_{on}$. The parameters of interest in the algorithms are: the jitter (guaranteed by our on-line algorithms as a function of the best possible off-line guarantee) and the buffer size (used by the on-line algorithm, as a function of the buffer size used by an optimal off-line algorithm).

Single delay jitter control

In this section, the best achievable delay jitter is analyzed. Then, an efficient off-line algorithm which attains the best possible delay jitter using a given buffer with space B is presented as in [1]. Then the main result of this chapter, which is an on-line delay-jitter control algorithm which attains the best jitter guarantee that can be attained by any (offline) algorithm which uses half the buffer space is presented as in [1]. Finally, a lower bound which shows that any on-line algorithm whose jitter guarantees are a function of the jitter guarantees of an off-line algorithm, must have at least twice the space used by the off-line algorithm is presented as in [1].

Off-Line Delay-Jitter Control

Let's start with the off-line case. Suppose the 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 delay jitter, using no more than buffer space. The off-line algorithm is defined as follows.

Algorithm A: Off-Line Delay-Jitter Control:

1. For each $0 \leq k \leq n$ define the interval

   $E_k = [a(k) - kX_a, a(k + B) - kX_a]$ (3.1)

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

2. Find an interval M of minimal length which intersects all intervals E_k.

3. For each packet k, let $P_k = \min(E_k \cap M)$, and define $s_{off}(k) = P_k + kX_a$.

Theorem 3.1: The sequence $\{s_{off}(k)\}_{K=0}^n$ is a non decreasing, B-feasible sequence with minimal delay jitter. The proof of this is presented in [1].

Delay bounds for offline Algorithm

Definitions:

Consider a sequence $\{a(k)\}_{K=0}^n$

of packet arrival times. Let $\{s_A(k)\}_{K=0}^n$

be the sequence of release times generated by algorithm A. Then we define the delay encountered by packet k to be

$d(k) = s(k) - a(k)$ (3.2)

Upper bound for the delay is defined as

$$d_{max}(k) = \max_{0 \le k \le n}\{s(k) - a(k)\} \quad (3.3)$$

Lower bound for the delay is defined as

$$d_{min}(k) = \min_{0 \le k \le n}\{s(k) - a(k)\} \quad (3.4)$$

From the algorithm discussed in section 3.1 the release sequence is defined as

$$s_{off}(k) = \begin{cases} minM + kX_a, & if\ minE_k < minM \\ a(k), & otherwise \end{cases} \quad (3.5)$$

Where

$$minM = \min_{0 \le k \le n}\{a(k+B) - kX_a\} \quad (3.6)$$

The delay bounds calculated for offline algorithm using ? are

when

$s_{off}(k) = a(k)$, the lower bound $d_{offmin} = 0$.

Similarly when $s_{off}(k) = minM + kX_a$, the upper bound $d_{offmax} = BX_a$.

On-Line Delay-Jitter Control

We now turn to our main result for delay-jitter control: an on-line algorithm using 2B buffer space, which guarantees delay jitter bounded by the best jitter achievable by an off-line algorithm using B space. The algorithm is simple: first the buffer is loaded with B packets, and when the $(B+1)^{th}$ packet arrives, the algorithm releases the first buffered packet. From this time onwards, the algorithm tries to release packet k after time kX_a. Formally, the algorithm is defined as follows.

Algorithm B: On-Line Delay-Jitter Control:

Define $s^*_{on}(k) = a(B) + kX_a \ \forall\ 0 \le k \le n)$.

The release sequence is defined by

$$s_{on}(k) = \begin{cases} s^*_{on}(k), & if\ a(k) \le s^*_{on}(k) \le a(k+2B) \\ a(k), & if\ s^*_{on}(k) < a(k) \\ a(k+2B), & if\ s^*_{on}(k) > a(k+2B) \end{cases} \quad (3.7)$$

Clearly, Algorithm B is an on-line algorithm. Theorem(as given in [1]): If, for a given arrival sequence, an off-line algorithm using space can attain delay jitter J, then the release sequence generated by Algorithm B has delay jitter at most J using no more than 2B buffer space. The proof of this is already presented in [1].

Delay bounds for online Algorithm

The release sequence is given by

$$s_{on}(k) = \begin{cases} s^*_{on}(k), & if\ a(k) \le s^*_{on}(k) \le a(k+2B) \\ a(k), & if\ s^*_{on}(k) < a(k) \\ a(k+2B), & if\ s^*_{on}(k) > a(k+2B) \end{cases} \quad (3.8)$$

where

$$s^*_{on}(k) = a(B) + kX_a \ 0 \le k \le n$$

We should note that the buffer size in this case is 2B where B is the buffer size used in off- line algorithm. Let the various slots in the buffer be 0 to 2B−1. Suppose packet B has arrived and the algorithm has started sending packets according to son(k) as described above. Delay is basically the amount of time the packet spends in the buffer.

The delay in the online case can be written as

$d_{on}(k) = s_{on}(k) - a(k)$

It is clear that if

$s^*_{on}(k) < a(k)$ then $d_{on}(k) = d_{onmin}(k) = 0$

If $a(k) \le s^*_{on}(k) \le a(k+2B)$ then it is clear that the buffer is not full. Suppose there are l(¡2B) packets already in the buffer. Then the kth packet will occupy the (l+1)th slot in the buffer. For this packet to have the maximum delay none of the l packets already present should be forced out of the buffer. That is to say, every packet should take maximum time to leave the buffer. The maximum time a packet in the 0th level can take is Xa, the maximum time a packet in the 1st slot can take is 2Xa and so on. So the maximum delay in this case will be (l + 1)Xa. Similarly if $s^*_{on}(k) > a(k+2B)$ then it can be shown that the maximum delay is $2BX_a$. So in the worst case $d_{onmax} = 2BX_a$ = (bufferspace) · X_a

Distributed delay jitter control:

In section 3, we have considered a single delay-jitter regulator. In this section, we prove an interesting property of composing many delay-jitter regulators employing our AlgorithmB. Specifically,

We consider a path of m links connecting nodes v_0, v_1, . . . , v_m, where v_0 is the source and is v_m the destination. We make the simplifying assumption that the propagation delay in each link is deterministic. We denote the event of the arrival of packet k at node j by a (k, j), and the release of packet k from node v_j by s (k, j). The input stream, generated by the source, is $\{s(k,0)\}_k$ (or $\{a(k,1)\}k$), and the output stream is $\{s(k,m)\}_k$. Each node has 2B/m buffer space, and for simplicity we assume that m divides B.

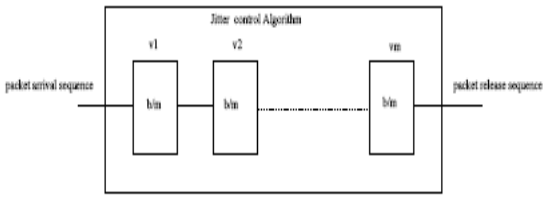

Figure 4.1: Jitter control model for distributed case

Distributed online delay jitter control

Algorithm BD: Distributed On-Line Delay-Jitter Control

For each node 1 ≤ j ≤ m, node v_j employs Algorithm B with buffer space 2B/m. Specifically, node sets $s^*_{on}(k,j)$ = a(B/m, j) + kX_a , and it releases packet as close as possible to $s^*_{on}(k,j)$ subject to 2B/m-feasibility (see Algorithm B). We prove that the jitter control capability of Algorithm BD is the same as the jitter control capability of a *centralized* jitter control algorithm with B total buffer space, under a certain condition for the beginning of the sequence (to be explained shortly). Put differently, one does not lose jitter control capability by dividing the buffer space along the path. The precise result is given in the theorem below.

Theorem 4.1: Suppose that for a given arrival sequence $\sigma = \{a(k)\}_{K=0}^{n}$ there exists a centralized off-line algorithm attaining jitter J using space B, with packet 0 released before time a(B/m). Then if σ is the release sequence of node v_o, the release sequence $\{S_{on}(k,m)\}_k$ generated by Algorithm BD at node v_m has delay jitter at most J. Intuitively, the additional condition is that there is a way to release the first packet relatively early by a centralized optimal algorithm. This condition suffices to compensate for the distributed nature of Algorithm BD. The condition is also necessary for the algorithm to work: if packets are input into the system at the start of the algorithm, then an off-line algorithm can still wait arbitrarily long before starting to release packets, while Algorithm BD is bound to start releasing packets even if only (2B/m) + 1 packets arrive.

The proof is essentially adapting the proofs of Algorithm B in chapter 3 to the distributed setting. We highlight the distinguishing points. Let the propagation delay over the link (v_{j-1}, v_j) be d_j and denote $D = \sum_{j=2}^{m} d_j$, the total delay of the on the path. The first lemma below bounds the desired release times of all packets at one node in terms of the desired release times in upstream nodes.

Lemma 4.2 : For all nodes 1≤ j≤ i≤ m and all packets k, $\{s_{on}(k,m)\}_k + \sum_{i=j+1}^{i} d_l \le \{s_{on}(k,i)\} \le \left(k + \frac{b}{m}(i-j), j\right) + \sum_{i=j+1}^{i} d_l$

Lemma 4.3 : if $s_{on}(k,m) > s^*_{on}(k,m)$ then $s_{on}(k,m)$ =a(k,1)+D

Lemma 4.4 : if $s_{on}(k,m) < s^*_{on}(k,m)$ then $s_{on}(k,m)$ =a (k+2B,1)+D

The proofs for the above lemmas are presented in [1].

End-to-End Delay bounds:

Consider a sequence $\{a(k,1)\}_{k=0}^{n}$ of packet arrival times at node '1'. Let $\{s_{BD}(k,m)\}_{k=0}^{n}$be the sequence of release times at node 'm' generated by algorithm BD. Then we define the delay encountered by packet k at node 'm' to be

$$d\_BD(k,m) = s_{BD}(k,m) - a(k,1) \quad (5.1)$$

Upper bound for the delay is defined as

$$d\_BD_{max}(k,m) = \max_{0\le k\le n}\{s_{BD}(k,m) - a(k,1)\} \quad (5.2)$$

Lower bound for the delay is defined as

$$d\_BD_{min}(k,m) = \min_{0\le k\le n}\{s_{BD}(k,m) - a(k,1)\} \quad (5.3)$$

Delay bounds for distributed offline algorithm:

From the information given in the section 4, the release sequence for distributed offline algorithm is defined as

$$s_{doff}(k,m) = \begin{cases} minM + kX_a & if\ \min E_k < minM \\ a(k,1) + D, & otherwise \end{cases} \quad (5.4)$$

Where

$$minM = \min_{0\le k\le n}\{a(k+B) + D - kX_a\}$$

$$= \min_{0\le k\le n}\{a(k+B) - kX_a\} + D \quad (5.5)$$

Where $D = \sum_{j=2}^{m} d_j$ is propagation delay over link.

When $s_{doff}(k,m)$=a (k+1) + D

$$d_{doff}(k,m) = s_{doff}(k,m) - a(k,1) \quad (5.6)$$

$$= a(k,1) + D - a(k,1) \quad (5.7)$$

$$= D \quad (5.8)$$

Since delay cannot be negative we have $d\_doff_{min} = D$ Similarly when $s_{doff}(k,m) = minM + kX_a$

$$d_{doff}(k,m) = s_{doff}(k,m) - a(k,1) \quad (5.9)$$

$$= minM + kX_a - a(k,1) \quad (5.10)$$

$$= \min_{0\le k\le n}\{a(k+B,1) - kXa\} + D + kXa - a(k,1)\} \quad (5.11)$$

$$= \min_{0\le k\le n}\{a(k+B,1) - (k+B)X_a)\} + D + BX_a - \{a(k,1) - kX_a\} \quad (5.12)$$

$$= \min_{0\le k\le n}\{a(k,1) - kX_a)\} + D + BX_a - \{a(k,1) - kX_a\} \quad (5.13)$$

The delay expression is the difference of constant $\min_{0\le k\le n}\{a(k,1) - kXa\} + BXa + D$ and a variable $\{a(k,1) - kXa\}$ dependent on k. This d(k,m) will be maximum for the minimum value of the variable. Equivalently, we can write

$$d\_doff_{max} = \min_{0\le k\le n}\{a(k,1) - kX_a)\} + D + BX_a - \min_{0\le k\le n}\{a(k,1) - kX_a\} \quad (5.14)$$

$$= Bx_a + D \quad (5.15)$$

Delay bounds for distributed online algorithm:

From the given information in the section 4, the release sequence for the distributed online algorithm is defined as

$$s_{on}(k,m) = \begin{cases} s^*_{on}(k,1) + D, & if\ a(k,m) \le s^*_{on}(k,m) \le a(k+2B,m) \\ a(k,1) + D, & if\ s^*_{on}(k,m) < a(k,m) \\ a(k+2B,1) + D, & if\ s^*_{on}(k,m) > a(k+2B,m) \end{cases} \quad (5.16)$$

Where $s^*_{on}(k,m) = a\left(\frac{B}{m},m\right) + kX_a \forall\ 0 \le k \le n$.

The delay in the distributed online case can be written as

$d_{don}(k,m) = s_{don}(k,m) - a(k,1)$.

It is clear that if $s^*_{on}(k,m) < a(k,m)$ then

$$d_{don}(k,m) = d\_don_{min}(k,m)$$

$$= a(k,1) + D - a(k,1)$$

= D (5.17)

Similarly if $s^{*}_{on}(k,m) > a(k+2B,m)$ then it can be shown that the maximum delay is 2BXa + D. So, in the worst case

d_don_{max} = $2BX_a + D$ (5.18)

= (bufferspace) · X_a + (propagationdelay)

Simulation Results

Distributed off-line case:

The distributed off-line and on-line algorithm was implemented in Matlab. We modeled the arrival sequence using Markov Modulated Poisson Process (MMPP). There are many fitting algorithms in the literature that take care of modeling internet traffic as MMPP.The total buffersize(B) is distributed to m nodes and checked the jitter.Each node has b/m buffer space.

The simulation parameters used for the simulation are

• n=200

• m=2

• Xa=3.04574995

We plot the jitter vs buffer size graph.

Distributed on-line case:

We had taken 200 arrival times for distributed offline algorithm. Now we consider 200 more (total 400) arrival times for the distributed on-line case using the same MMPP model

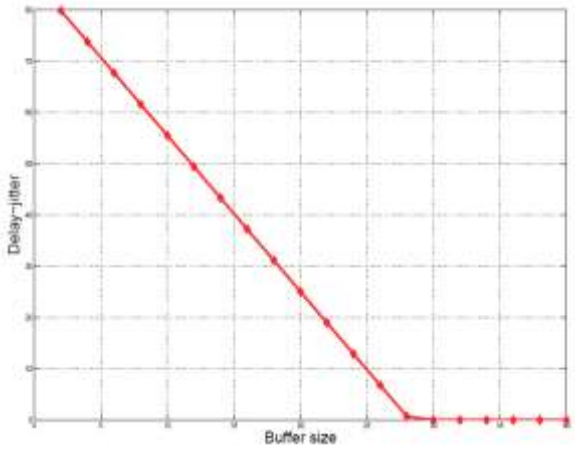

Figure 6.1: distributed off-line jitter for various buffer size

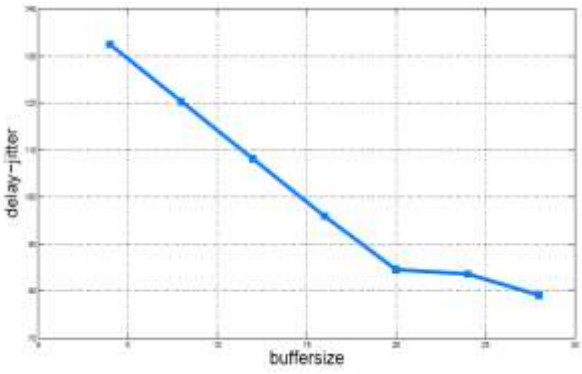

Figure 6.2: distributed on-line jitter for various buffer size

Used before. We keep the average inter arrival time Xa=3.04574995.

The results were as expected. As buffer size increases both distributed on-line and offline jitter decrease linearly. Figure 7.1 shows the jitter vs buffer graph for distributed on-line (with n1= 400) and Figure 7.2 shows the graph for distributed off-line (with n = 200 inputs). Note here on-line jitter is more than the off-line case. This is because of the extra arrival points considered in the on-line case.

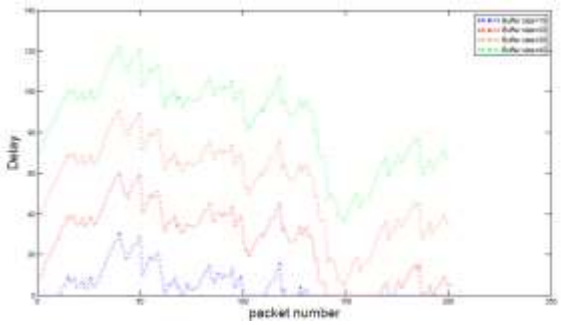

Figure 6.3: Delay for various packet numbers

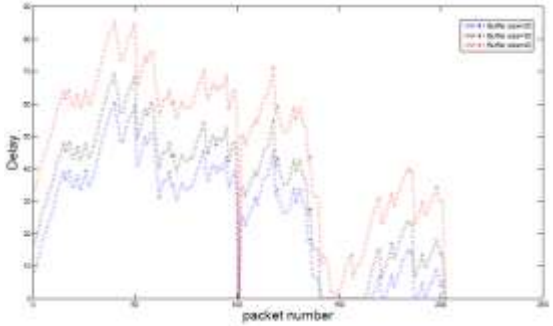

Figure 6.4: Delay for various packet numbers

Conclusion

The delay bounds will increase incase of distributed delay jitter control algorithms than the non-distributed case because propagation delay comes in to picture incase of distributed one. . The delay jitter get same for both distributed and non distributed algorithms. . Incase of distributed online algorithm b < b_ case is happening at earlier buffersize than the centralized online algorithm. The delay bounds calculated mathematically are checked by simulation.

References

[1] Jitter Control in QoS Networks, Yishay Mansour and Boaz Patt-Shamir, IEEE/ACM transactions on networking, vol. 9, no. 4, pages 492-502, august 2001

[2] Delay -jitter control in multimedia applications,Sumathi Kadur, Forouzan Golshani and Bruce Millard,Springer Berlin / Heidelberg, Volume 4, Number 1, pages 30-39, February 1996

[3] Design and applications of a delay jitter control scheme for packet-switching internetworks, Domenico Ferrari, Springer Berlin / Heidelberg, Volume 614, pages 69-83,

Ramesh Chandra Kothapalli,born in A.P,India.Completed M.Tech from IIT Madras. B.Tech from R V R & J C College of Engineering, Guntur.Presently working as Asst.Prof.At Vishnu Institute of Technology,Bhimavaram.He has a Teaching Experience of three years. His Research work interests in Communications

Mr.A Krishna Chaitanya Varma,born in A.P,India.Completed M.Tech from K L C E,Vaddeswaram.B.E from S R K R Engineering College,Bhimavaram. Presently working as Asst.Prof in Vishnu Institute of Technology,Bhimavaram.His Research interest in Communications

Mr ch.v.v.s.srinivas born in A.P india, Completed M.Tech,B.E from S R K R Engineering College,Bhimavaram. Presently working as Asst.Prof in Vishnu Institute of Technology ,Bhimavaram his research interest in communications

Mr. G.prasannakumar born in A.P, India completed M.Tech from JNTU college of Engineering, Hyderabad, B.E from S.R.K.R Engineering college, Bhimavaram .presently working as Asst.Proff in Vishnu Institute of Technology,Bhimavaram, His research interest in signal processing