ISSN 2278-3091

**International Journal of Advanced Trends in Computer Science and Engineering**, Vol.2 , No.1, Pages : 407 - 412 (2013)
*Special Issue of ICACSE 2013 - Held on 7-8 January, 2013 in Lords Institute of Engineering and Technology, Hyderabad*

# JITTER CONTROL USING SLIP OPTIMIZED RATE JITTER CONTROL ALGORITHM

**V.Indupriya**
**Belcy D Mathews**
Indupriya2006@gmail.com, mathews.d.belcy@gmail.com,
**L I E Technology, JNTU H, Hyderabad**

**ABSTRACT**

In packet-switched [7]networks, quality of service is affected by various factors. The problems when travelling from the origin to destination they are Low throughput, Dropped packets, Errors, Latency, Jitter[4-6], Out-of-order delivery. Its applications are in Streaming media, VoIP, Videoconferencing, Online games, Industrial control systems protocols[8].

A novel and efficient algorithm has been designed which overcomes the inaccuracies of the algorithms existing in literature. Slip Optimized Rate Jitter Control[1][2] Algorithm using a buffer size of 2B which reduces rate jitter[4-6] drastically when compared to the on-line algorithm. Also, derived the mathematical relationship to find δ based on parameters such as buffer size (B), number of packets in the buffer (j) and average inter-departure time ($X_{avg}$). The on-line algorithm uses a buffer of size 2B+h for any $h \geq 1$ and comparison of its jitter to the jitter[4-6] of an optimal off-line algorithm using buffer size B. It proves that this algorithm guarantees the difference is bounded by a term proportional to B/h. The jitter control[1][2] algorithm is that reduces the rate jitter for a given buffer size B.

***Keywords: Jitter, Off-line algorithm, online algorithm, BER, delay***

## I. INTRODUCTION:

In today's high-speed digital communications industry, timing variations in a signal have become more of an issue due to a need to maximize signal BW, while maintaining signal integrity. These timing variations are called jitter. Large systems contain many high-speed products, a jitter 'budget' must be maintained in order to minimize the probability of incurring a bit error. Bit errors are referred to in terms of BER. Controlling jitter in the digital signal path plays a very important role in minimizing BER and increase system reliability. These systems transfer data at various rates based upon industry standards. When frequencies increases its data rates should increase, so does the demand for lower jitter.

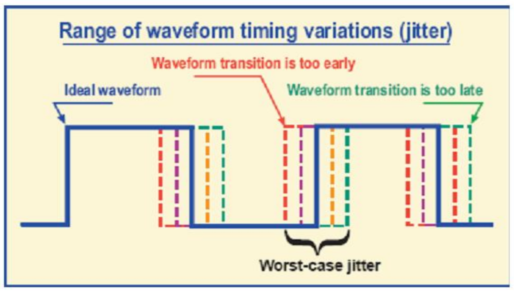
Range of waveform timing variations (jitter)
Waveform transition is too early
Ideal waveform
Waveform transition is too late
Worst-case jitter

**Fig. 1. Waveform with Jitter**

Errors can occur due to a reduction in the sampling window that the clock signal uses to decipher whether the data is at 1 or 0. As the frequency of the data signals increase, jitter consumes more and more of the sampling window.

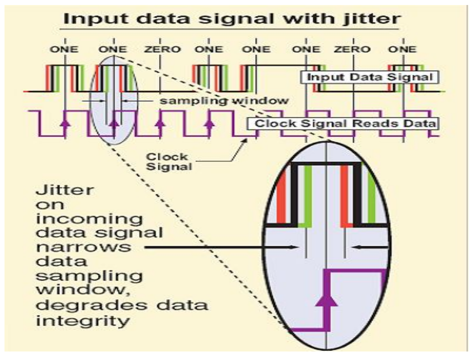
Input data signal with jitter
ONE ONE ZERO ONE ONE ONE ZERO ONE
Input Data Signal
sampling window
Clock Signal Reads Data
Clock Signal
Jitter on incoming data signal narrows data sampling window, degrades data integrity

**Fig.2. Data Sampling Window Reduction1**

ISSN 2278-3091

When multiple data signals are being transmitted simultaneously on a PCB board, a data signal may be affected by adjacent signals are being transmitted at high speeds[9]. This is called crosstalk.

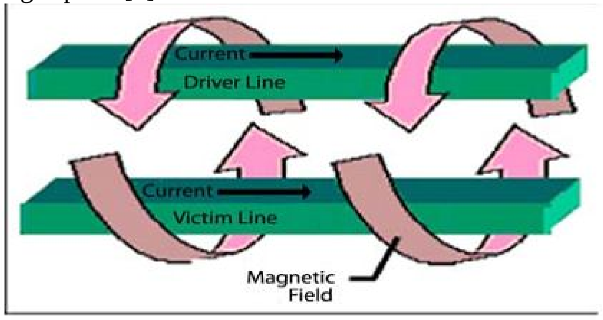

**Fig3.. Crosstalk Coupling Between Adjacent Signals**

In order to minimize these effects, one needs to create space between the traces that carry these signals. By placing low frequency and energy carrying signals such as a select trace to a multiplexer or an equalizer level select trace next to the signal in question will help to minimize any coupling. Data transmission is done through the use of differential signals. It provides shielding from external coupling since both the true and the complement signals are coupled the same way. The interference is masked because noise coupled to these lines is common mode to the differential pair. Electromagnetic interference can affect a signal through nearby power pins or power supplies. When a power supply signal switches, a significant amount of energy from a magnetic field can be coupled into the signal path. It create an unwanted current that can alter a signals bias point as well as its amplitude.

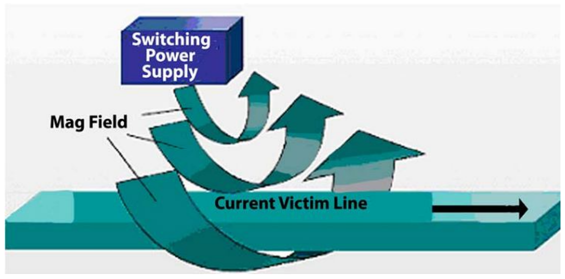

**Fig4. Power Supply Switching Interference**

To minimize this, all power pins are placed as far as possible from the data signals. All power supplies must be properly bypassed with a capacitor or through other means of shielding. As mentioned before, differential data paths also reduce the effects of power supply switching since the effect is common mode to both differential lines. Another source of data degradation is reflections or SW due to the proper termination of the signal line. Since reflections are due to the signal itself, it consider DJ contribution to jitter. Signals suffering from reflections often have spikes or humps throughout the waveform[10] that can alter its rise and fall times and thus, can contribute to incorrect sampling of the data bits.

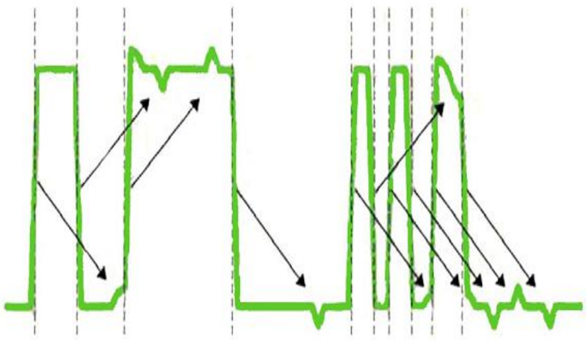

**Fig5. Signal Integrity Affected By Improper Line Termination**

To minimize this, it is important to terminate all data signals with their appropriate termination schemes. As speeds and the desire for improved performance increase, the amount of jitter in a system or product will become more of a concern. Information is extracted from serial data streams by sampling the data signal at specific instants. The presence of jitter changes the edge positions with respect to the sampling point. An error will then occur when a data edge falls on the wrong side of a sampling instant. The result of the summation of deterministic and random jitter is another probability distribution. The distribution plots probability against timing error magnitude.

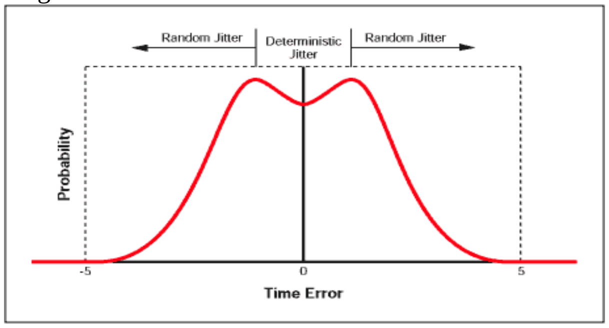

**Fig. 6 The shape of distribution shown is referred to as a bimodal response**.

The probability of a data error associated with the sampling instant is the sum of the probabilities that either the first data transition will arrive too late or the second data transition will arrive too early. Jitter buffers are used to change asynchronous packet arrivals into a synchronous stream by turning variable network delays into constant delays at the destination end systems. Jitter buffer underflows and overflows cause voice quality degradation. Adaptive jitter buffers aim to overcome these issues by dynamically tuning the jitter buffer size to the lowest acceptable value. Advanced formulas can be used to arrive at network-specific design recommendations for jitter. The data is stored in a buffer as it is retrieved from an input device or just before it is sent to an output device. The data stored in a data buffer are stored on a physical storage media. A majority of buffers are implemented in software, which typically use the faster RAM to store temporary data, due to the much faster access time compared with hard disk drives. Buffers used in a FIFO ..

**Delay and delay-jitter:**

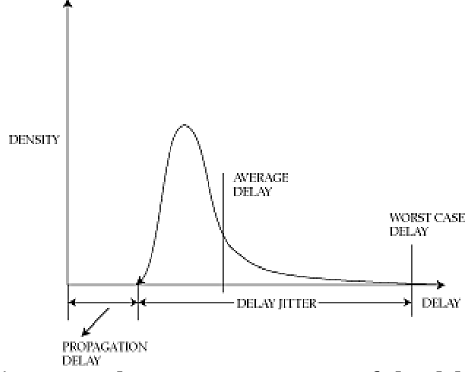

**Fig 8.Bound on some parameter of the delay distribution curve**

1. Two components: regulator and scheduler
2. Incoming packets are placed in regulator where they wait to become eligible
3. Then they are put in the scheduler
4. Regulator shapes the traffic.

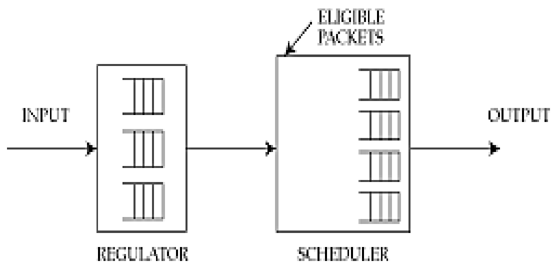

**Fig 9.jitter regulator and link scheduler**

## II. Mathematical Backgroud

### i)Network congestion

In data networking and queuing theory, network congestion occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queuing delay, packet loss or the blocking of new connections.

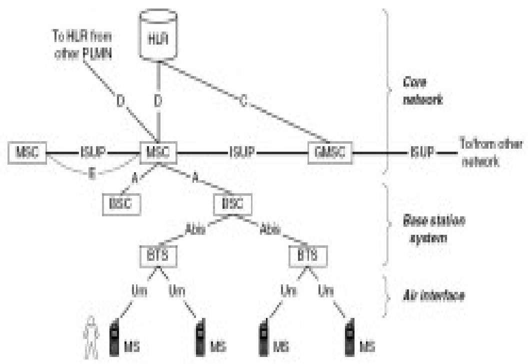

**Fig 10. GSM Network Architecture**

GSM Network Entities are MS, BTS, BSC, MSC, VLR, HLR, EIR, MSC.

**ii) The buffer model**

Let us consider the following abstract communication model for a node in the network as shown in Fig. 3.1. 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 is governed by a jitter control[1][2] 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.

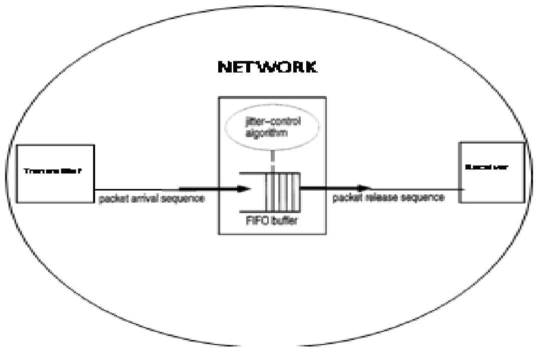

**Fig 11 Jitter control model. The jitter-control algorithm controls packet release from the buffer, based on the arrival sequence.**

Jitter control algorithms[3], 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. The release time sequence generated by algorithm A using a buffer of size B must satisfy the following condition for all 0≤k≤n:
$a(k) \leq s_A(k) \leq a(k+B)$

Where $a(k) = \infty$ for $k \geq n$.
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 (3.1). 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. A times sequence is a non-decreasing sequence of real numbers. The superscript $\sigma$ has been omitted when the context is clear. The average rate of is simply 1/$X_A^\sigma$.The main concern here is about the rate jitter of $\sigma$, 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 $\sigma$ is defined to be

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

The means for analyzing the performance of jitter-control algorithms is competitive analysis [1]. Release times which minimize jitter may require knowledge of the complete arrival sequence in advance. The results are expressed in terms of the performance of on-line algorithms using buffer space $B_{on}$ where usually $B_{off} < B_{on}$. The main interest here is in two parameters of the algorithms: the jitter and the buffer size The problem of minimizing the rate-jitter is considered, i.e., how to keep the rate at which packets are released within the tightest possible bounds. An on-line algorithm for rate-jitter control using space 2B+h is presented and compared with an off-line algorithm using space B and guaranteeing jitter J. Our algorithm guarantees rate jitter at most J+c.B/h, for some constant c It is also shown how to obtain rate jitter which is a multiplicative factor from optimal, with a simple modification of the algorithm. Without having knowledge of the exact average inter-arrival time, jitter guarantees will come into effect after an initial period in which packets may be released too slowly. It is also shown that without doubling the space, no guarantees in terms of the optimal rate-jitter can be made. As an aside, it is remarked that off-line rate-jitter control can be solved optimally using linear-programming technique.

## III. RATE JITTER CONTROL ALGORITHMS

### i)Off-line Rate Jitter Algorithm

1. For each 0≤k≤n, define the interval

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

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

**ii) On-line Rate Jitter Algorithm**

The algorithm is specified with B buffer size of an off-line algorithm, i.e., $B_{off} = B$; h≥1 space parameter for the on-line algorithm, such that $B_{on} = 2B + h$; $I_{min}, I_{max}$ bounds on the minimum and maximum inter-departure time of an Off-line algorithm, respectively.

$X_{avg}$Average inter-departure time in the input (and also the output) sequence.

The parameters $I_{max}$ and $I_{min}$ can be thought of as requirements; these should be the worst rate-jitter bounds the application is willing to tolerate. The goal of a rate-jitter control algorithm is to minimize the rate jitter, subject to the assumption that space B is sufficient (for an off-line algorithm) to bound the inter-departure times in the range $[I_{min}, I_{max}]$. A trivial choice for $I_{min}$ and $I_{max}$ is $X_{min}$ and $X_{max}$, which are the minimal and maximal inter-arrival times in the input sequence. However, using tighter $I_{min}$ and $I_{max}$, one may get a much stronger guarantee. The jitter guarantees will be expressed in terms of B, h, $I_{min}$, $I_{max}$, $X_{avg}$ and J, the best rate jitter for the given arrival sequence attainable by an off-line algorithm using space B.

Note that for an on-line algorithm, achieving rate jitter of even $I_{min} - I_{max}$ may be nontrivial. These are bounds on the performance of an off-line algorithm, whose precise specification may depend on events arbitrarily far in the future.

The basic idea in this algorithm is that the next release time is a monotonically decreasing function of the current number of packets in the buffer. In other words, the more packets there are in the buffer, the lower the inter-departure time between the packets (and thus the higher will be the release rate).

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

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

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$\mathrm{IDT}(j) \le X_{avg}$.
- Let $= \min\{j|\ \mathrm{IDT}(j) \le X_{avg}\}$, 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,$t \le \mathrm{last\_departure} + \mathrm{IDT}(j)$, where is the number of packets currently in the buffer, then a packet is delivered and last departure is updated.

Let J be the best rate-jitter attainable (for an off-line algorithm) using buffer space B for a given arrival sequence. Then the maximal rate-jitter in the release sequence generated by Algorithm B is at most$J + (I_{max} - I_{min})(2B+4)/h$.

The no. of packets in the buffer is never more than B+2 buffer slots away from the slots which correspond to rates generated by an optimal off-line algorithm. We now formally analyze Algorithm B. Fix an optimal execution of the off-line algorithm. Let us denote the maximum and minimum inter-departure times of the off-line execution by $Y_{max}$ and$Y_{min}$, respectively. (Hence, the jitter attained by the off-line algorithm is $Y_{max} - Y_{min}$). With these quantities, we also define the following terms:

$$U = \min\{j|\mathrm{IDT}(j) \le Y_{min}\},\ I_U = \mathrm{IDT}(U)$$
$$L = \max\{j|\mathrm{IDT}(j) \ge Y_{max}\},\ I_L = \mathrm{IDT}(L).$$

Note that$L \le S \le U$. We shall also use the following shorthand notation. Let $B_{on}(t)$ and $B_{off}(t)$ denote the number of packets stored in time in the buffers of the Algorithm B and of the off-line algorithm, respectively, and let $\mathrm{diff}(t) = B_{on}(t) - B_{off}(t)$ , i.e., how many packets does the Algorithm B has more than the off-line algorithm at time t. We use extensively the following trivial property of the difference.

1. For all t,$-B \le \mathrm{diff}(t) \le B_{on}(t)$.
2.For any two time point $t_1 \le t_2$ $\mathrm{diff}(t_2) = \mathrm{diff}(t_1) - S_{on}(t_2,t_1) + S_{off}(t_2,t_1)$
.
3. For all times t, $\mathrm{diff}(t) \le U + 1$.
4.For all times $t > t^*$, $\mathrm{diff}(t) \ge (L-1) - B$.

**iii) Multiplicative Rate Jitter Algorithm**

The ratio between the maximal and minimal inter-arrival times. This measure is called the multiplicative rate jitter, or m-rate jitter for short.

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

For $\delta = (I_{min} - I_{max})^{1/h}$.

The maximal m-rate-jitter in the release sequence generated by Algorithm B using function $\mathrm{IDT}_m$ is at most $J.(I_{max}/I_{min})^{(2B+4)/h}$.

**iv) TOLERANCE BASED JITTER CONTROL ALGORITHM**

**4.1System Model**

Poisson times sequence a (n) which represents a series of data packets arriving at the input of the buffer and let $X_{avg}$ be the mean time of the arrival sequence. The arrival time of a given k$^{th}$ packet is represented by a (k). All the packets are assumed to be of equal sizes which are released in FIFO order. Let s (k) denote the time at which k$^{th}$ packet is released by the algorithm.

**v.Efficient Rate-Jitter Control Algorithm**

**Algorithm C:** Rate Jitter Control

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

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

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. These packets are not released until the time, when the

ISSN 2278-3091

**International Journal of Advanced Trends in Computer Science and Engineering**, Vol.2 , No.1, Pages : 407 - 412 (2013)
*Special Issue of ICACSE 2013 - Held on 7-8 January, 2013 in Lords Institute of Engineering and Technology, Hyderabad*

number of packets stored in the buffer becomes equal to B. This is known as the buffer loading stage. $s(1) = a(B)$.
The first packet is sent at the ame time the B$^{th}$ packet arrives. The remainder of the algorithm follows the following condition:

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

where $\delta = (B - j)\frac{X_{avg}}{\beta}$,

j is the no. of packets in the buffer at a given instant of time and $\beta$ is a constant which should always be greater than B+1. $\delta$ depends on the difference between constant B, variable j and$\beta$. $\delta$ is designed in such a way that the rate jitter of the system is reduced. $\delta$ varies being positive, negative or zero.
Suppose, when $a(B + k) > a(B) + k.X_{avg}$
$s(k) = a(B) + (k-1)X_{avg} + \delta$
Similarly, $s(k+1) = a(B) + k.X_{avg} + \delta$
When the number of packets in the buffer (j) becomes equal to B (i.e., j=B), then $\delta$ becomes zero.
This implies $s(k+1) = a(B) + k.X_{avg}$
$= a(B) + (k-1)X_{avg} + X_{avg}$
$= s(k) + X_{avg}$
In this case, s(k+1) becomes independent of $\delta$ 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 $\delta$ close to zero. $\delta$ becomes zero when the number of packets in the buffer is equal to B as proved earlier. Therefore, in order to keep $\delta$ close to zero Maintain the no. of packets in the buffer to be close to B.
If j > 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. There should not be any overflow in the buffer. S
If j < B, then the packets are released at a relatively slow rate. By this, there should not be any underflow at the buffer. The value of $\beta$ in the denominator part of $\delta$ should be greater than B+1.

## IV.CONCLUSIONS

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.Delay of the on-line algorithm has not constant after the some buffer size compare to off-line algorithm. But rate jitter of the on line algorithm is reduced compare to off line algorithm. In off line algorithm the buffer overflow &under flow problem has been faced. Also, multiplicative rate jitter which is approximately equal to on-line rate jitter algorithm with a small change in IDT (j) has been explained. 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 .

## V. 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].Analysis of jitter control algorithm in Qos networks, Jagadish.S,Manivasakan.R, IEEE transactions on networking, 29 Dec 2011.
[3] Wavecrest Corp, “Jitter Fundamentals”, SMPB-00019 Rev.1,
[4] Hancock, Johnnie, “Jitter- Understanding it, measuring it, eliminating it; Part 3: Causes of Jitter, Copyright 2004, Summit Technical Media, LLC,
[5] Nehring, Dan, “Oscillator Jitter FAQ”,
[6] Wavecrest Corp, “Understanding Jitter”, Copyright 2001,
[7] N. R. Figuera and J. Pasquale, “Leave-in-Time: A new service discipline for real-time communications in packet-switching networks,” in Proc. ACM SIGCOMM, 1995, pp. 207–218.
[8] R. Händel, M. N. Huber, and S. Schröder, ATM Networks: Concepts, Protocols, Applications, 3rd ed. Reading, MA: Addison-Wesley, 1998.
[9] C. R. Kalmanek, H. Kanakia, and S. Keshav, “Rate control servers for very high-speed networks,” in Proc. GLOBECOM ’90, 1990, pp. 12–20.
[10] Wavecrest Corp, “Understanding Jitter”, Copyright 2001.