IETE Technical Review

ISSN: 0256-4602 (Print) 0974-5971 (Online) Journal homepage: http://www.tandfonline.com/loi/titr20

IP Telephony and Delay Jitter Control—An Overview

Aman Kansal & Abhay Karandikar

To cite this article: Aman Kansal & Abhay Karandikar (2003) IP Telephony and Delay Jitter Control—An Overview, IETE Technical Review, 20:4, 289-296, DOI: 10.1080/02564602.2003.11417086

To link to this article: http://dx.doi.org/10.1080/02564602.2003.11417086

Published online: 26 Mar 2015.

IETE Technical Review
Vol 20, No 4, July-August 2003, pp 289-296

IP Telephony and Delay Jitter Control — An Overview

AMAN KANSAL AND ABHAY KARANDIKAR

Department of Electrical Engineering, Indian Institute of Technology, Bombay, Mumbai 400 076, India.

Packet network based telephony has become an attractive application because of many advantages it offers in terms of costs and services. This article discusses some of the major technical challenges in the implementation of IP telephony. One of the challenges in IP telephony is that of delay jitter in the received audio. The problem of jitter in delays needs to be handled even in the newer networks having high bandwidth and low delays. In this paper, we discuss the jitter problem in detail and provide an overview of various methods used to tackle it. These include a new method proposed by the authors which offers performance gains in terms of packet loss and jitter over existing methods.

1. INTRODUCTION

IN recent years, the use of packet networks for voice has increased very rapidly. Packet telephony, also called Internet Protocol (IP) telephony, has multiplied its traffic by more than an order of magnitude every year for the past few years. This is due to the fact that the provisioning of telephony over Internet has many advantages. The first is the cost advantage for consumers. The second major advantage occurs to corporate users who have to maintain two separate networks for voice and data. If the packet switched network employed for data can be used for voice as well, it will mean one less network to maintain.

Packet telephony is not without its challenges and there are still open problems that need to be addressed. Since the best effort packet networks like the Internet do not offer any explicit quality of service guarantees in terms of packet loss, end to end delay and delay jitter, the fluctuations in network load can severly degrade the voice quality.

In this paper, we address one of these problems, namely delay jitter. The delay jitter is defined as the difference between the largest and the smallest delay suffered by packets on the connection. This can also be thought of as the width of the end to end delay histogram. Specifically, we review some mechanisms to address the problem of degradation in voice quality due to delay jitter. Delay jitter can occur even if the network is not heavily congested and the end to end to delay and packet losses are within tolerable limits. This is due to different queueing delays experienced by different packets.

We begin this paper by outlining some major challenges involved in packet telephony. In the next few sections, we review delay jitter control algorithms. We conclude by discussing the effect of pushing jitter into silence zones.

2. CHALLENGES IN IP TELEPHONY

We first briefly provide an architecture of IP telephony. The International Telecommunication Union (ITU) has worked out a standard for packet network based telephony, called the H.323 [1]. This standard allows interoperability of packet network based phones and the conventional public switched telephone network (PSTN). The basic architecture of the system is shown in Fig 1. An alternative signaling mechanism is provided by Session Initialization Protocol (SIP) [2] designed by the Internet Engineering Taskforce (IETF), which may be used instead of the H.323.

The gateway shown in the figure is responsible to connect the H.323 packet network to the circuit switched network, such as the Public Switched Telephone Network (PSTN). The multipoint controller unit is responsible for handling conference calls. The system also has another entity called the gatekeeper, which takes care of call admission control and mapping of phone numbers to IP addresses. The H.323 voice terminals, also referred to as H.323 endpoints, have an audio interface and carry out audio compression, packetization and the related network layer tasks. A network

Paper No 80-B;

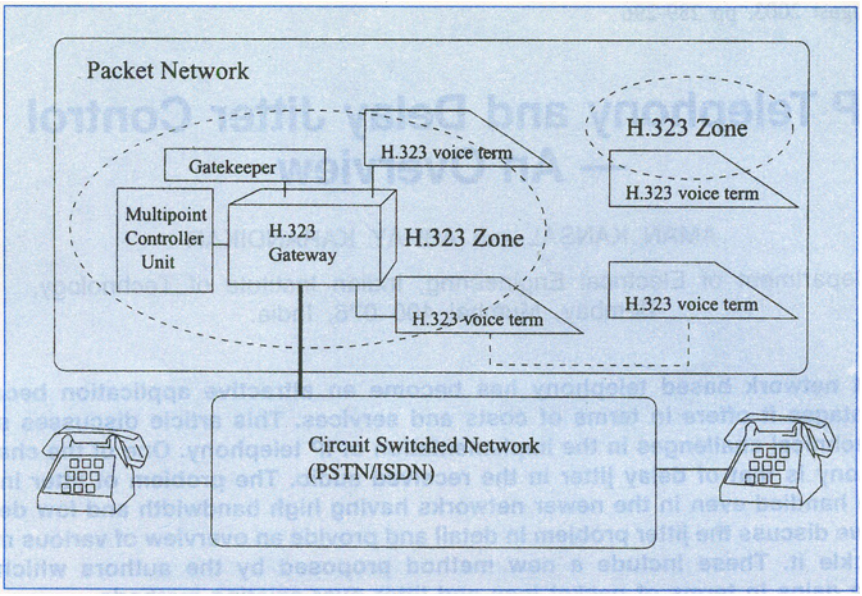


Fig 1 Packet telephony architecture

of H.323 endpoints, connected to the same gateway is referred to as an H.323 zone. Calls can be established between two H.323 endpoints on the same H.323 zone, on different zones or between an H.323 endpoint and a PSTN telephone. Calls can also be made from an H.323 endpoint not connected to any H.323 zone to another H.323 endpoint. An H.323 voice terminal collects the digitized voice at regular intervals, say every 20 ms (other time intervals could be used depending on the voice coding). These chunks are given to the real time transport protocol (RTP).

Real Time Protocol (RTP) defines an encapsulation mechanism that allows multimedia systems to transmit and receive real-time media using the best-effort packet delivery provided by IP. RTP also timestamps the transmitted packets, which, as we show later, is useful for the playout of audio packets at the receiver. Along with RTP, there is a control protocol called Real Time Control Protocol (RTCP). The RTCP protocol runs in parallel with RTP and provides data delivery monitoring, that is, knowledge about the condition of the underlying network. The applications may make use of the information provided by RTCP in various ways including that of adapting the source rate to the changing network bandwidth. The RTP packets are encapsulated in UDP packets and then sent over IP based networks.

The system would work well if the network carries the audio traffic without any significant delay, delay jitter and packet loss. Such is not the case, however, for IP networks. The media encapsulation layer thus has to take special measures to achieve an acceptable quality of played out voice.

Different approaches have been suggested to tackle these problems. To mitigate the effects of loss, methods based on introducing redundancy in the audio stream have been suggested [2,3] - these involve sending extra forward error correction (FEC) packets. For instance, a redundant packet, obtained by exclusive OR-ing of n packets is sent after every n packets and the receiver can use it to reconstruct upto one lost packet among the n packets. Another method, used in FreePhone, transmits two parallel streams, one at high quality and another at low quality. The low quality stream, which is also a lower data rate stream, is to be used when the high quality stream is congested. Apart from redundancy, the receiver also uses interpolation to fill up the gaps caused by lost packets [4].

If the receiver starts playing out packets as soon as the first packet arrives, all other packets, which arrive with a delay longer than that of the first one, will be dropped as playout has to be at regular intervals. The audio stream can tolerate some packet loss depending on the voice codec used. Packet loss upto 5% is typically considered to be tolerable for both raw voice and G.711 coded voice with loss repair. The receiver can buffer the received packets before playout to smoothen out the effects of network delay jitter (Fig 2). However, buffering introduces additional delay. The challenge is therefore to balance the tradeoff between the delay and packet loss. The rest of the paper discusses this problem in detail.

3. JITTER REDUCTION AT THE RECEIVER

The end-to-end delay for a packet consists of the

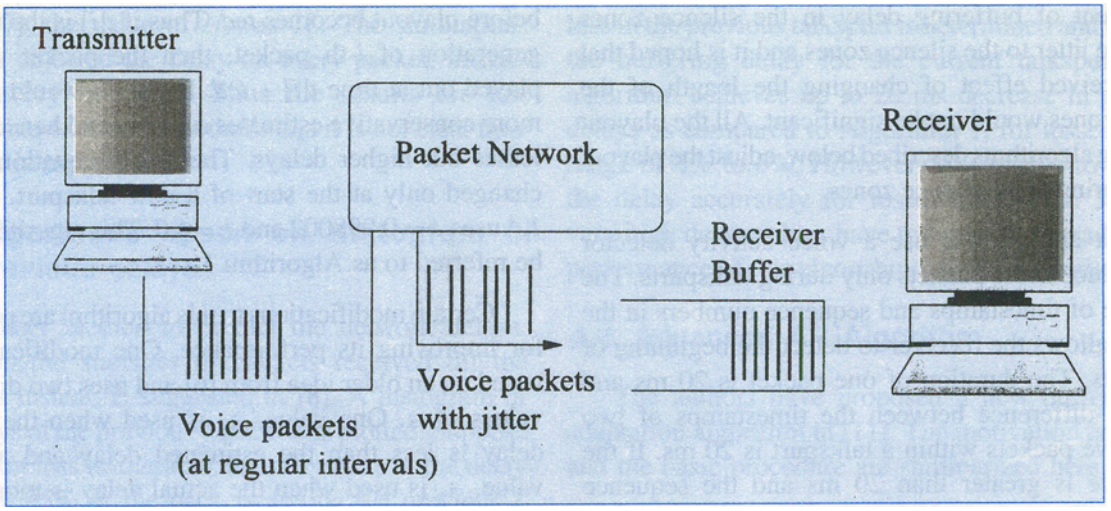


Fig 2 Buffering packets at receiver to tackle network delay jitter

time taken by the packet to reach the receiver, i.e, the propagation delay, queuing delay within the network and the time the packet has to wait in the buffer at the receiver before it is played out, i.e, the buffering delay. The end-to-end delay is an important parameter in the design of an interactive voice application. The value of this delay should be ideally below 150 ms, however, tolerable interactivity is maintained up to 400 ms. Beyond that, the delays seriously impair the interactivity and the conversation may become irritating to the user.

The tradeoff between and delay and loss in buffering is depicted in Fig 3. The left curve shows the times at which packets are sent from the sender. The time intervals between successive packets are regular. The curve on the right shows the times at which the packets are received and the jitter is visible.

The effects of the delay jitter without incurring any packet loss could be mitigated by starting the playout at time t_1, as shown in the figure. The value of t_1 may turn out to be very large in practical scenarios because some packets in the audio stream may be lost or may be suffering very long delays and this may render it impossible to achieve interactivity. However, as mentioned earlier, since audio streams can sustain losses of a few packets without an appreciable degradation in voice quality, playout may be started earlier than t_1, say at t_2 (Fig 3). The problem is now to chose a value of the end-to-end delay to be used for an audio stream which keeps packet loss below an acceptable limit specified by the application, typically around 5%. The buffering delay for a particular packet will then be the end-to-end delay minus its network delay.

The first challenge is of course to estimate this end-to-end delay for the specified loss. Minimum possible delay must be used to achieve the best possible perceived quality. Once estimated, this delay may be kept fixed for the entire duration of the call. However there is a problem with this approach because the network conditions can change over relatively short periods and the same delay may not meet the loss criterion for the entire call. To overcome this problem a buffering algorithm is required which dynamically adjusts the end-to-end delay by adapting to the changing network conditions. Many such algorithms have been proposed including a very recent one by the authors. These will be described in the next section.

Since the audio packets must themselves be played out periodically, any adjustment must be done during the silence period within the audio stream. The

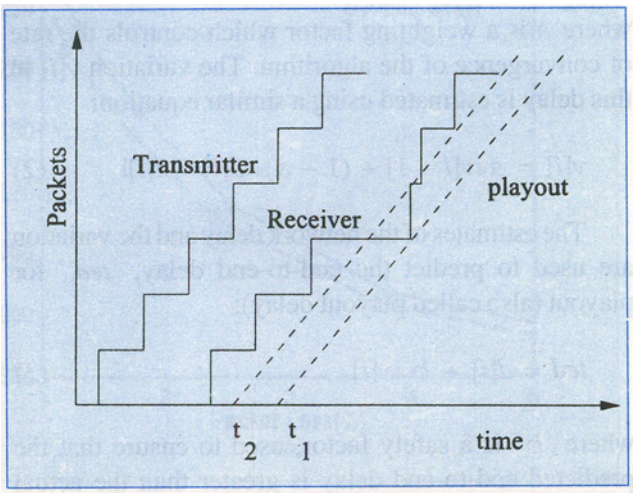


Fig 3 The playout buffering problem

adjustment of buffering delay in the silence zones shifts the jitter to the silence zones and it is hoped that the perceived effect of changing the length of the silence zones would not be significant. All the playout buffering algorithms described below, adjust the playout delay during these silence zones.

Most applications use a voice activity detector and produce audio packets only during talkspurts. The presence of timestamps and sequence numbers in the packets allows the receiver to detect the beginning of talkspurts. The duration of one packet is 20 ms and thus the difference between the timestamps of two successive packets within a talkspurt is 20 ms. If the difference is greater than 20 ms and the sequence numbers are consecutive, the receiver infers that a new talkspurt has begun. If the sequence numbers are not consecutive, the receiver can verify that the difference is due to loss of some packets in the network.

4. BUFFERING DELAY ADJUSTMENT ALGORITHMS

Various algorithms have been studied in the literature for the purpose of delay adjustment. We discuss some of them.

4.1. Autoregressive (AR) Estimate Based Algorithms

The basic algorithm (used in audio conferencing tools NeVot 1.4 and Vat), uses an AR estimate based method to estimate the average network delays and the variations in the delay [5]. The observed network delay $n[i]$ for the i-th packet is used to update the estimate of the network delay $d[i]$ as per the following equation:

$$d[i] = a\, d[i-1] + (1-a)\, n[i] \qquad (1)$$

where a is a weighting factor which controls the rate of convergence of the algorithm. The variation $v[i]$ in this delay is estimated using a similar equation:

$$v[i] = a*v[i-1] + (1-a)*|d[i] - n[i]| \qquad (2)$$

The estimates of the network delay and the variation are used to predict the end-to-end delay, *'ted,'* for playout (also called playout delay):

$$ted = d[i] + b*v[i] \qquad (3)$$

where 'b' is a safety factor, used to ensure that the predicted end-to-end delay is greater than the actual network delay for any packet with a high probability. Each packet is buffered such that its end-to-end delay before playout becomes *ted*. Thus if $t[i]$ is the time of generation of i-th packet, then the packet will be played out at time $t[i]$ + *ted*. Higher b would lead to more conservative estimates of delay and hence lower losses but higher delays. The *ted* in equation (3) is changed only at the start of a new talkspurt. NeVot 1.4 uses $a = 0.998002$ and $b = 4.0$. This algorithm will be referred to as Algorithm 1.

Certain modifications to this algorithm are possible for improving its performance. One modification is based on an older idea from [6] and uses two different values of a. One value, a_1 is used when the actual delay is less than the estimated delay and another value, a_2 is used when the actual delay is more than the estimated delay. In [6] $a_1 = 0.9375$ and $a_2 = 0.75$ are suggested. The playout delay and variation are estimated as per equations (1) and (2) using these two separate a's. Thus, the algorithm has greater flexibility and may have better performance as it takes into account the network conditions.

Another modification, (from NeVot 1.6) is to record the delays of all the packets in the most recent talkspurt and use the minimum of these delays as an estimate of the network delay in the next talkspurt. The variation in delay and the final end-to-end delay are estimated as before.

Sudden and large increases in the network delay can sometimes occur and are followed by a series of packets arriving simultaneously. These are referred to as delay spikes. Algorithm 1 and its variants cannot respond fast enough to a spike as spikes cause a discontinuity in the observed delay which cannot be tracked by an autoregressive estimate. According to a modification [5] the usual AR delay estimation is abandoned during a spike. The delay estimate is set equal to the most recent delay observed and that value is used to decide the buffering delay. This algorithm will be referred to as Algorithm 2 in subsequent sections.

A performance comparison of these algorithms is presented in [5] and it is found that none of the methods guarantees improved performance in all circumstances. The spike detection method for instance performs better than Algorithm [1] on certain traces but worse on others as shown in [5].

4.2. Adaptive Filter Based Algorithms

An adaptive algorithm based on the Normalized Least Mean Squares (NLMS) filter is proposed in [7]. In this method, the delays are predicted using an adaptive filter. This filter uses the previous N delays to predict the current delay and the filter coefficients are updated as per the NLMS equation.

In [7], the value of N was 18. The simulations however adjusted the delay on every packet, instead of on every talkspurt. Thus the results are not comparable with other methods and it is not clear how the jitter is suppressed.

4.3. Algorithms based on Histogram of previous delays

Another method to predict the network delays, based on the statistics of packets received till the previous instant, is suggested in [8]. A histogram of the delays of the previous L packets is plotted. Suppose 5% packet loss is tolerable, then the value of the delay such that 95% of the packets among the previous L packets arrived with a delay lower than this value, is determined from the histogram. This value is then used as the predicted value of delay for the next talkspurt. In [8], L = 10, 000 is used. The algorithm offers some gains on most traces, especially in regions of packet loss below 1%.

A variation to this algorithm is suggested in [9], where, instead of taking just the last L packets, a gradual aging procedure is applied to the histogram being plotted. The older packet delays are given lower weightage than the more recent ones.

Another variation to the histogram approach is suggested in [10]. Instead of using the last L packets to plot the histogram, the delays of the packets in the most recent talkspurt only are used. From this, the buffering delay that would have provided the desired loss in the previous talkspurt is determined and used as the buffering delay for the current talkspurt. The algorithm achieves up to 20 ms decrease in playout delays as compared to Algorithm 1, for losses in the range of 3% to 6%. However, it is unable to predict the delay accurately for losses lower than 3% and very high delay values have to be used, degrading the performance of this algorithm in low loss regions.

4.4. Advanced AR Algorithm

The authors have proposed a new buffer delay adaptation algorithm in [11]. The motivation behind it and the basic procedure are summarized here.

In equation (1), the choice of a affects the significance of the most recent data in calculating the AR estimate of average network delay $d[i]$. When the algorithm was tested with various values of a on different traces it was found that small variations in the value of a affect how rapidly the average estimate is changed by variations in the actual delays, that occur due to change in network conditions. These variations influence the predicted playout delay in equation (3). Hence the delay loss tradeoff depends significantly on the value of a. An improvement in the performance can be expected if the value of a is adjusted adaptively to suit the network conditions. This is the aim of the proposed algorithm.

The algorithm maintains two separate values of the weighting parameter a in equation (1) a_1 and a_2. Suppose $a_2 > a_1$. Here, a_1 is used for the actual

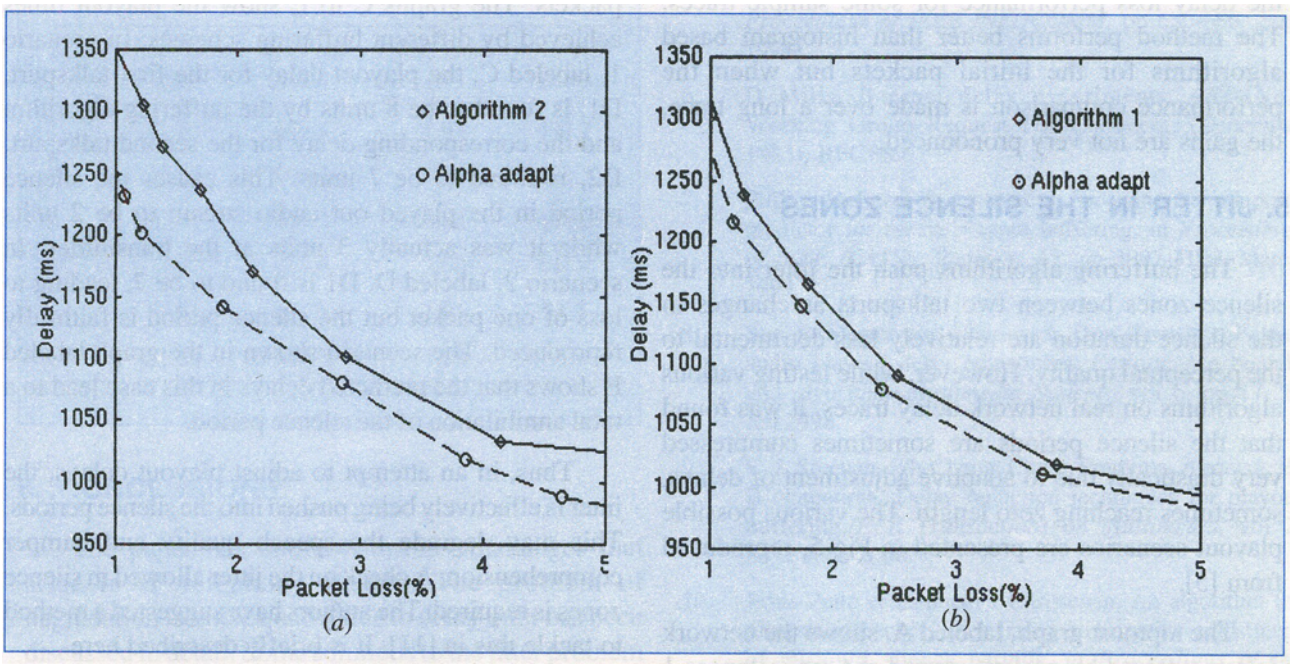


Fig 4 Performance advantages of a-adaptive algorithm over (*a*) a algorithm 2 (both methods with spike detection) (*b*) a algorithm 1 (both methods without spike detection) on traces between North America and Japan

playout and a_2 for simulating the playout. If a_2 produces better results in the simulations, a_1 is set equal to a_2 and hence, the better value is used for playout. Also a_2 is increased further to move towards an optimal value. If however at any stage, a_2 gives poorer results than a_1, the algorithm starts reducing a_2 and the same procedure is repeated in the opposite direction of changing a_2. This is referred to as the a adaptive algorithm. Its performance was compared to Algorithms 1 and 2 through simulations in [11] on various wide area network traces collected between locations in North America and Europe and between North America and Japan, taken from [5]. While comparing the performance of the advanced AR method with that of Algorithm 2, spike detection was added to the advanced AR method too. Figure 4 shows the end-to-end delays incurred by the buffering algorithms for values of packet losses up to 5%. It can be seen that the a adaptive algorithm reduces the delays and hence will lead to enhanced voice quality in an IP telephony application.

4.5. Hybrids

The histogram based algorithms require the knowledge of the past L packet delays. This information is not available at the beginning of the call and for the first few packets, the estimate based on a very small value of L gives poor performance. So, [12] suggests a hybrid algorithm, in which Algorithm 1 or its variants are used until L packets have been received and then the delays of these L packets can be used for computing the histogram. The simulations in [12] show gains in the delay loss performance for some sample traces. The method performs better than histogram based algorithms for the initial packets but when the performance comparison is made over a long trace, the gains are not very pronounced.

5. JITTER IN THE SILENCE ZONES

The buffering algorithms push the jitter into the silence zones between two talkspurts as changes in the silence duration are relatively less detrimental to the perceptual quality. However, while testing various algorithms on real network delay traces, it was found that the silence periods are sometimes compressed very drastically due to adaptive adjustment of delays, sometimes reaching zero length. The various possible playout scenarios are presented in Fig 5, reproduced from [5].

The topmost graph, labeled A, shows the network delays suffered by packets started at time instances 1 to 6 and 9 to 13. These periods represent two talkspurts. No packet starts at time instances 7 and 8 and this

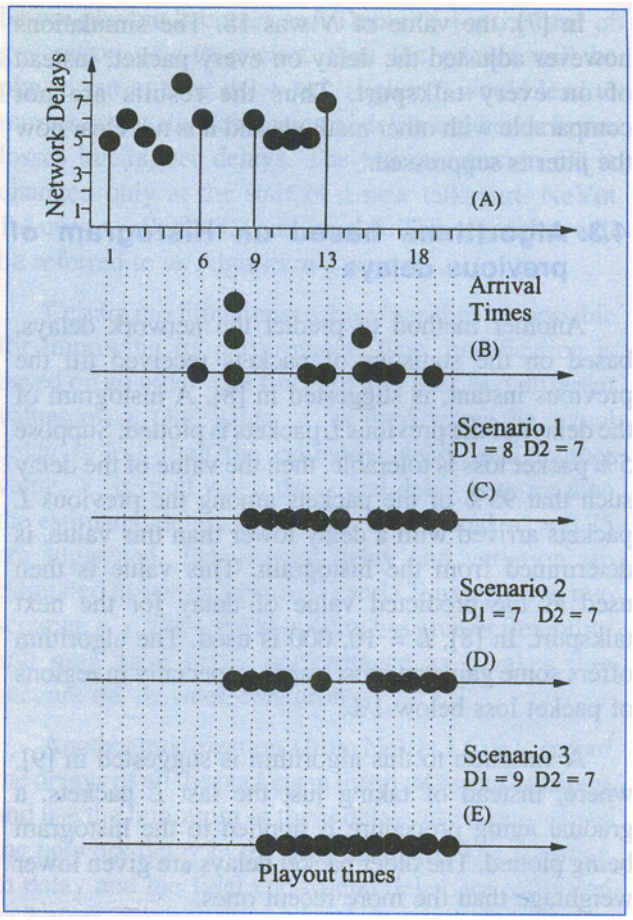


Fig 5 The playout jitter problem due to adaptive delay adjustment [4]

period forms a silence zone. The second graph, labeled B, shows the arrival times at the receiver for the packets. The graphs C to E show the playout times achieved by different buffering schemes. In scenario 1, labeled C, the playout delay for the first talkspurt, D1, is found to be 8 units by the buffering algorithm and the corresponding delay for the second talkspurt, D2, is found to be 7 units. This causes the silence period in the played out audio stream to be 2 units while it was actually 3 units at the transmitter. In scenario 2, labeled D, D1 is found to be 7, leading to loss of one packet but the silence period is faithfully reproduced. The scenario shown in the graph labeled E shows that the predicted delays in this case lead to a total annihilation of the silence period.

Thus, in an attempt to adjust playout delays, the jitter is effectively being pushed into the silence periods. This may degrade the speech quality and hamper comprehension. A check on the jitter allowed in silence zones is required. The authors have suggested a method to tackle this in [11]. It is briefly described here.

The change in *ted* is made when the first packet of a new talkspurt arrives, based on an estimate from

the previous data. However, at this stage, since the last packet of the older talkspurt and the first packet of the new talkspurt have both been received, it is possible to calculate the actual silence duration between the two talkspurts using the transmitter timestamps of these packets. Also, the estimated playout time that is intended to be used by the algorithm for starting the playout of the first packet in the new talkspurt is known. The actual playout time of the last packet of the previous talkspurt can be subtracted from that intended playout time to calculate the silence duration in the played out audio stream. Here, if the silence period is being compressed to below a certain tolerance, say to below 50% of the original, the playout time of the first packet is delayed by an appropriate time duration such that the silence duration in the played out sequence is at least 50% of the true silence duration.

The effect of this procedure on the delays and losses is presented in table 1 for six different audio traces for a tolerance of 50% based on Algorithm 1. The new loss and *ted* refer to the packet loss and end-to-end delay achieved when the jitter control procedure is applied and the values labeled old are the ones obtained without this procedure. It can be seen that the jitter control procedure does not increase the delay significantly. Thus it may be used to preserve silence zones to a reasonable tolerance, for preventing excessive deterioration of the voice quality.

TABLE 1 Effect of jitter control procedure on loss and delay

Trace	New Loss (%)	New *ted* (ms)	Old Loss (%)	Old *ted* (ms)
1	1.96	606.1	1.97	597.7
2	0.85	905.4	0.90	896.5
3	0.82	449.6	0.82	448.0
4	2.99	147.6	3.00	144.4
5	1.01	91.7	1.01	91.7
6	1.19	924.0	1.19	915

6. CONCLUSIONS

The paper has outlined various challenges that arise in IP telephony system. The problem of degradation in voice quality due to delay jitter has been discussed in detail. Some solutions to the jitter problem and a generic procedure to force a bound on the jitter have been reviewed. This includes a new algorithm called *a* adaptive algorithm. The algorithms have been compared in terms of delay and loss performances. It has been observed that the *a* adaptive algorithm performs better than existing algorithms showing large gains on delay in the low loss region. Specifically, the *a* adaptive algorithm has been found to reduce the playout delay by up to 100 ms for packet losses less than 2%. This is a significant improvement for interactive audio. However, the differences in their performances in terms of perceptual quality need to be determined through measurement of subjectivity (MOS) tests. This could be actively pursued as future research. Experimental study is also required on new Internet delay traces collected in different types of networks with different load conditions.

REFERENCES

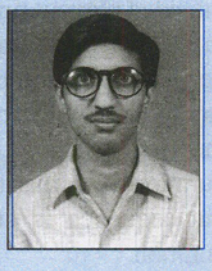
1. Packet-based multimedia communication systems, ITU.T. Recommendation H.323: *http://www.itu.int/rec/recommendation.asp*, 1998.
2. N Shacham & McKenney, Packet recovery in high speed networks using coding and buffer management, in *Proceedings of IEEE INFOCOM,* San Fransisco, CA, May 1990, pp 124-31.
3. Jean-Clirysostome Bolot & Andres Vega Garcia, The case for FEC based error control for packet audio in internet, *ACM Multimedia Systems,* 1997.
4. N S Jayant, Effects of packet loss on waveform coded speech, in *Proceedings of the Fifth Int. Conference on Computer Communications,* Atlanta, GA, October 1980, pp 275-280.
5. R Rimjee, Jim Kurose, Don Towslet & Henning Schulzerine, Adaptive playout mechanisms for packetized audio applications in wide area networks, *Proceedings of IEEE INFOCOMM,* pp 680-686, June 1994.
6. D Mills, Internet delay experiments, *ARPANET* Working Group Request For Comment, (December 1983), RFC 889.
7. Phillip DeLeon & Cormac J Sreenan, An adaptive predictor for media playout buffering, in *Proceedings of IEEE ICASSP,* Phoenix, Az, pp 3097-3100, March 1999.
8. Sue B Moon, Jim Kurose & Don Towsley, Packet audio playout delay adjustment: Performance bounds and *algorithms,Multimedia Systems,* vol 6, pp 17-28, Jan 1998.
9. C J Sreenan Jyh-Cheng Chen, Prathima Agrawal, & B Narendran, Delay reduction techniques for playout buffering,*IEEE Transactions on Multimedia,* vol 2, no 2, June 2000.
10. Jesus Pinto & Kenneth J Christenen, An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods, in *Proceedings of the 24th Conference on Local Computet- Network,* Lowell, Massachusetts, 7-20 October 1999.

11. Aman Kansal & Abhay Karandikar, Adaptive delay estimation for low jitter audio over internet,in *Procceding of IEEE Globeco,* San Antonio, USA, November 2001.

12. Harri Marjamaki & Raimo Kantola, Performance evaluation of an IP voice terminal, *in Proceedng of the Fifth IFIP Conference on Intelligence in Networks,* Asian Institute of Technology, Thailand, November 1999.

Authors

Aman Kansal received his BTech in Electrical Engineering and MTech in Communications and Signal Processing from Indian Institute of Technology Bombay in 2001 and 2002 respectively. He is currently pursuing a doctorate at University of California, Los Angeles. Aman has been an active researcher in the area of voice over Internet, wireless networks and pervasive computing. He was a recipient of the Microsoft Innovation Award in 2001 for prototyping a mobile telephony system for airplanes (team effort) and several prizes in technology development at the national level. He was awarded the NTSE scholarship by the Govt. of India, and the graduate research fellowship by the Regents of the University of California.



Abhay Karandikar received his MTech and PhD degrees from IIT Kanpur in 1988 and 1994 respectively. During 1988-89, he worked in Indian Space Research Organization, Ahemedabad. During 1994-97, he worked in Center for Development of Advanced Computing, Pune as Team Coordinator in High Speed Communications Group. Since 1997, he is working in IIT Bombay where currently he is an Associate Professor in the department of Electrical Engineering. Dr Karandikar has consulted extensively for industries in the area of communications network. His research interests include Quality of Service in Internet, VLSI in Communications Systems and Statistical Communications Theory.