Adaptive Delay Estimation for Low Jitter Audio over Internet

Aman Kansal and Abhay Karandikar
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Bombay, India-400076

Abstract— **Real time voice applications typically produce uniformly spaced voice packets and faithful reconstruction demands that these be played out at the same intervals. Best effort packet networks, however, produce variable delays on different packets and the receiver is required to buffer the received packets before playout. Excessive buffering delays deteriorate the system performance for interactive audio and so intelligent algorithms that keep this delay minimum while maintaining an acceptable packet loss have to be employed. In this research, we develop a new "α-adaptive" algorithm which offers considerable reduction in delays compared to existing algorithms, especially for low packet losses. A generic jitter control procedure is also proposed which may be used with any buffering algorithm to enhance its jitter performance without significantly affecting the delay loss tradeoff. Further, an existing algorithm based on Normalized Least Mean Squares filter is discussed and modifications are proposed for its practical implementation. All suggestions are supported by simulations on internet delay traces.**

I. Introduction

The promise of toll quality voice over Internet has spawned a host of research activities. The main problems in audio over packet networks arise due to delays and jitters in delays. Improvements can be made at the transmitter, in the network and at the receiver, see [1].

Here we investigate the adjustments that can be made at the application layer in the receiver to reduce jitter. Received packets are buffered at the receiving site and their playout delayed in order to compensate for the network jitter. The buffering delay is adaptively adjusted while at the same time keeping the packet loss as low as possible. Packet losses up to 5% are considered acceptable for human conversation [2]. However, changing the playout delay in between a stream will cause jitter and hence adjustments are made only in the silence periods between two talkspurts in the audio stream. We propose a new playout algorithm, a jitter control procedure and modifications to an existing Normalized Least Mean Squares filter based algorithm.

The next section describes the playout problem in some detail. Section III summarizes buffering techniques currently used or proposed and comments on their performance. Section IV describes our playout techniques. Related simulations are presented in Section V. Section VI concludes the paper.

II. The adaptive Playout Problem

The main objective of the playout algorithm is to minimize the delay experienced while maintaining an acceptable packet loss. There should ideally be no jitter in the playout.

A possible sequence of generation of voice packets at the transmitter is shown in Figure 1, the left curve. The curve on the right shows packet reception. Delays between subsequent packets are not uniform as the network delays are variable and the need for buffering is apparent. A lossless playout without jitter will involve storing the packets such that the packet with the largest delay is received in time for playout. This could be achieved by starting the playout at time t_1, in Figure 1. The dashed line shows the playout time at the receiver. In this method, excessive delay for even one packet will require the receiver to chose a very large value of t_1. This delays all packets unnecessarily as some loss could have been tolerated. If instead the playout is started at time t_2, the end to end delay is reduced but packets that arrive after their playout times have to be dropped. Efficient algorithms which can chose a small t_2 and keep loss acceptable are required.

Buffering techniques based on a fixed buffer delay set at the start of audio session are unable to achieve satisfactory audio quality ([3], [4]). Most algorithms adjust t_2 throughout the audio stream as the behavior of network delays changes and higher and lower values of buffering delay may be required to keep a low loss. The adjustment is usually made during a silence zone in the audio stream and it is expected that the expansion or compression of the silence zone will not cause a significant degradation of the perceived quality.

III. Review of Current Delay Adjustment Algorithms

A. Autoregressive (AR) Estimate Based Algorithms

The basic algorithm (used in audio conferencing tools NeVot 1.4 and Vat) as described in [3], estimates the average packet delay d_i using:

$$d_i = \alpha d_{i-1} + \{1-\alpha\} n_i \tag{1}$$

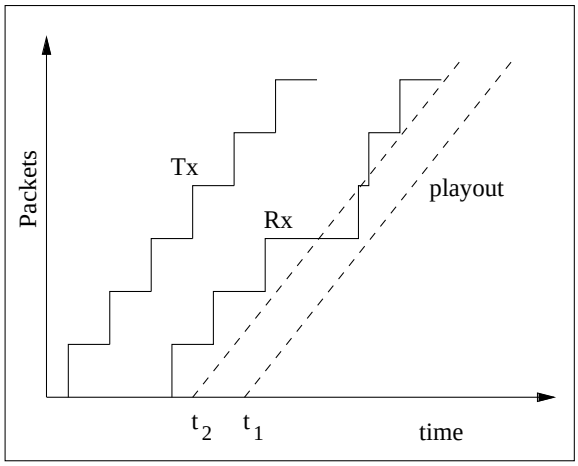

Fig. 1. Playout Jitter Problem

0-7803-7206-9/01/$17.00 © 2001 IEEE

where n_i is the delay suffered by the ith packet in the network and α is a weighting factor which controls the rate of convergence of the algorithm. The variation v_i in this delay is estimated as:

$$v_i = \alpha v_i + \{1-\alpha\}|d_i - n_i| \quad (2)$$

This is used to fix the end to end delay *ted*, for playing out the next packet as:

$$ted = d_i + \beta v_i \quad (3)$$

Here, β is a safety factor, used to ensure that the predicted delay is greater than the actual delay with a high probability. The *ted* in (3) is changed only at the start of a new talkspurt. NeVot 1.4 uses $\alpha = 0.998002$ and $\beta = 4.0$. This algorithm will henceforth be referred to as Algorithm 1.

One modification, suggested in [3] is to check for delay *spikes*-sudden and large increases in the network delay followed by a series of packets arriving simultaneously. During a spike, the AR method described above is abandoned and the buffering delay is based only on the most recent delays observed. This algorithm will be referred to as Algorithm 2.

A performance comparison of these algorithms is presented in [3] and it is found that none of the methods can uniquely be declared the best.

B. An Adaptive Filter Based Algorithm

An algorithm based on the Normalized Least Mean Squares (NLMS) filter is proposed in [4]. The delays are predicted using an NLMS adaptive filter:

$$h_{i+1} = h_i + \frac{\mu}{\{n_i^T n_i + a\}} n_i e_i \quad (4)$$

where h_i is the $N \times 1$ vector of adaptive filter coefficients, μ is the step size, n_i is the $N \times 1$ vector containing the most recent N network delays and e_i is the estimation error given by:

$$e_i = h_i^T n_i - \hat{n} \quad (5)$$

where $\hat{n}$ is the estimate of next delay obtained by filtering n_{i-1} through h_i. The playout delay is calculated as in (3) but $\hat{n}$ is used instead of d_i. This delay is updated on every packet in [4].

C. Algorithms based on Histogram of previous delays

Another method to predict network delays is suggested in [5] based on the statistics of packets received till the previous instant. A histogram of the previous L packets is plotted and a value higher than the delay of a chosen percentage of packets is used as the predicted value for the next talkspurt. In [5], $L = 10,000$ is used. Two variations have been proposed in [6] and [7]. A hybrid method is explored in [8].

A bound on the minimum playout delay achievable for a given packet trace has been calculated in [5] offline, after recording its packet arrival times. Simulations in [5] reveal that there is some gap between the playout delay achieved by the existing algorithms and the minimum bound, especially in regions of low loss, under 2%. Hence it is of interest to improve the playout buffering algorithms to achieve lower delays in the low loss regions.

The buffering algorithms adjust the delay during the silence zones between two talkspurts as changes in the silence duration are relatively less detrimental to the perceptual quality. However, we have observed that the silence periods are sometimes compressed drastically, reaching zero length. Thus, in an attempt to adjust playout delays, the jitter is effectively being pushed into the silence periods. This not only degrades the speech quality but may also hamper comprehension. A check on the jitter allowed in silence zones is required.

The NLMS algorithm described in [4] has adjusted the delays on a per packet basis. This does not solve the problem of jitter in the playout. Further, its delay loss performance is not much better than that of Algorithm 1. Our simulation results illustrate that on traces from [5], with talkspurt based delay adjustment, it is in fact worse. The parameters μ and filter order N in (4) are preset in the algorithm and might not be suitable for different network traces. The choice of the filter order N will depend on the correlation properties of the delays and the choice of μ may be affected by the variance in the delays.

IV. Proposed Algorithms

Based on the above comments, we investigate the following issues:

A. The Delay Loss Performance

In (1), the choice of α affects the significance of the most recent data in calculation of the AR estimate of average network delay d_i. By experimenting with differnet values of α we observed that small changes in its value can significantly affect the delay loss tradeoff. This is because the value of α decides how rapidly d_i tracks the changes in the network delay behaviour.

An improvement in the performance can be expected if the value of α is adjusted adaptively so that an optimal value can be reached after a few packet arrivals.

We propose the following algorithm to achive this:

---

1. $\alpha = 0.998002$, $increment = 0.0001$,
   $\alpha_2 = \alpha - increment$
2. Calculate ted based on α and ted_2
   based α_2 using (1),(2) and (3)
   Use ted_2 for actual playout
3. loss1=Loss(L,α), loss2=Loss(L,α_2)
   /*Loss(X,Y) calculates
   the loss based on $\alpha = Y$ in the previous
   X talkspurts.*/
4. If loss2 < loss1
   if $\alpha_2 < \alpha$ then $\alpha_2 = \alpha - increment$
   if $\alpha_2 > \alpha$ then $\alpha_2 = \alpha + increment$
5. If loss2 > loss1
   if $\alpha_2 < \alpha$ then $\alpha_2 = \alpha + increment$
   if $\alpha_2 > \alpha$ then $\alpha_2 = \alpha - increment$
6. Repeat steps 3 to 5 every L^{th}
   talkspurt.

Playout delay is adjusted on every talkspurt. This is referred to as the α-*adaptive* algorithm. Here *increment* is much smaller than α, as very small changes in α lead to large changes in the loss and playout delays.

B. Jitter in the Silence Periods

The delay adjustment pushes the jitter into the silence periods and simulations reveal that some silence zones are totally annihilated. We propose a procedure to avoid this.

The change in *ted* is made when the first packet of a new talkspurt arrives. 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 measure the actual silence duration between the two talkspurts using the transmitter timestamps. Also, from the calculated playout time which is intended to be used by the algorithm for starting the playout of the first packet and the playout time of the last packet of the previous talkspurt, the estimated silence duration in the played out audio can be obtained. Here, if the silence period compression is below a certain *tolerance*, say 50% of the original, the playout time of the first packet is delayed by an appropriate amount to achieve at least the silence period set by the tolerance. This procedure is referred to as the *jitter control procedure* in the subsequent sections.

If the predicted playout expands the silence period, the procedure may force the playout to begin so as to keep the silence zone within the specified tolerance, but this will lead to increased packet losses. This part has not been implemented by us.

C. NLMS Implementation and Modification

The NLMS algorithm described in [4] has adjusted the playout delays on a per packet basis. This should obviously be changed to per talkspurt basis. The parameter μ and filter size N were preset. It may be appropriate to estimate the statistics of the delay trace from the first few packets received and use these to decide the values of the above parameters.

The performance of the NLMS filter can be improved if the data fed to it is decorrelated before passing through the filter. This will lead to faster convergence and lower errors. We suggest using Discrete Wavelet Transform (DWT) for decorrelation. The order of the decorrelating filter depends on how far into the past the correlation extends in the data, which may be estimated from the covariance lags,

$$r_N(k) = \frac{1}{N} \sum_{j=0}^{N-k-1} n_{j+k} n_j \qquad k = 0, 1, ..., N/2 \qquad (6)$$

where n_i is the network delay for the ith packet and N is the number of delay samples used.

V. SIMULATIONS

We have used the internet traces from [5], (Table I) for our simulations. This facilitates easier comparison of the proposed algorithms with those suggested by other authors.

The traces used in [4] were also obtained to verify our NLMS implementation by matching with their results. But these don't have talkspurts marked out and can be used only for per packet delay adjustment.

TABLE I
WIDE AREA NETWORK TRACES USED

Trace	Sender	Receiver	Time, Day	Duration (seconds)
1	UMass	Osaka	0035,Fr	649
2	UMass	GMD	1105,Fr	1040
3	UMass	GMD	0841,Tu	1384
4	UCI	INRIA	2100,Sa	1091

A. Experiments on α-adaptive Algorithm

The α-adaptive algorithm has been compared with Algorithm 1 and its variant Algorithm 2 for losses up to 5%, the acceptable limit for voice. We use $L = 5$ and $increment = 0.0001$. The α-adaptive algorithm performs better than both Algorithm 1 and Algorithm 2, showing large gains on delay in the low loss regions on most traces. On others, it performs at least as good. Changing L did not significantly affect the performance, while making α larger lead to higher losses on some of the traces.

Figure 2 compares the α-adaptive algorithm with Algorithm 2 on trace 1. It can be seen that the α-adaptive algorithm reduces the playout delay by up to 100ms for low packet losses (less than 2%). This is a significant improvement for interactive audio since it leads to a two way end to end delay reduction of up to 200ms. While comparing with Algorithm 2, spike detection is also added to α-adaptive.

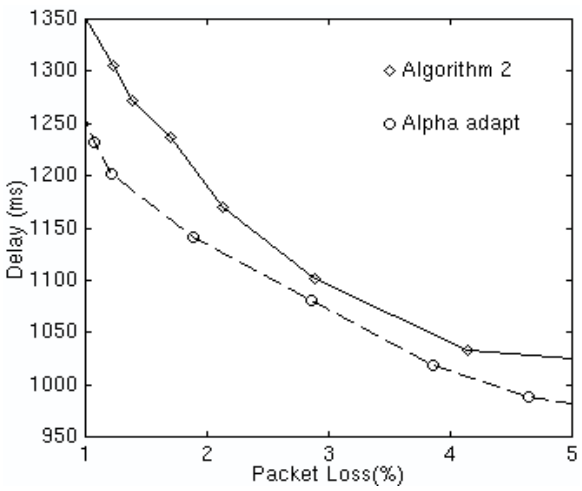

Fig. 2. Delay-loss comparison of α-adaptive algorithm with Algorithm 2.(Trace 1)

Figure 3 compares α-adaptive with Algorithm 1. Improvements are notable, especially in low loss regions.

Next, the same comparisons are made on Trace 2 (Figures 4 and 5). Again, the α-adaptive algorithm performs better.

Comparisons are also made with Traces 3 (Figure 6) and 4 (Figure 7). It can be seen that α-adaptive performs at least as

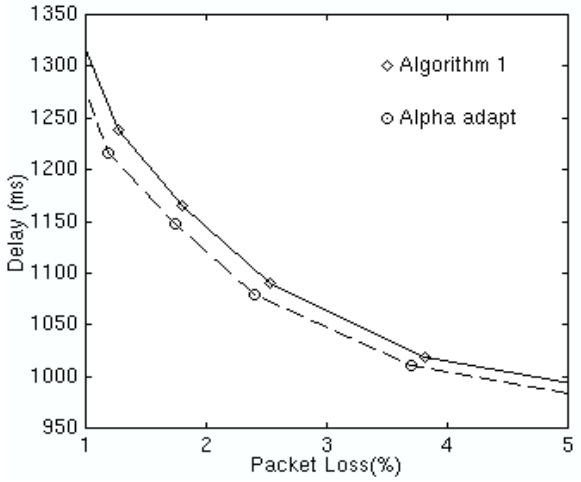

Fig. 3. Delay-loss comparison of α-adaptive algorithm with Algorithm 1.(Trace 1)

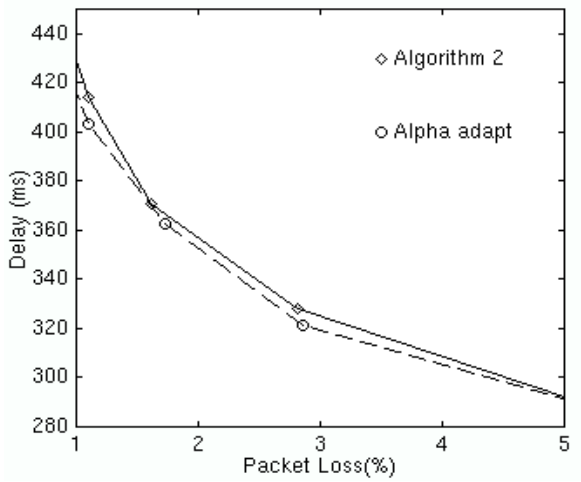

Fig. 4. Delay-loss comparison of α-adaptive algorithm with Algorithm 2.(Trace 2)

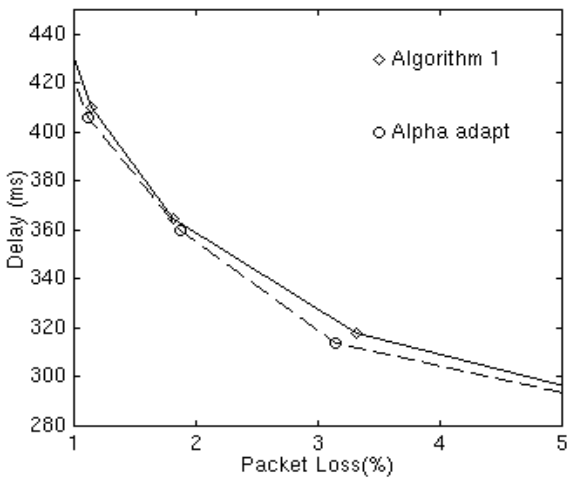

Fig. 5. Delay-loss comparison of α-adaptive algorithm with Algorithm 1.(Trace 2)

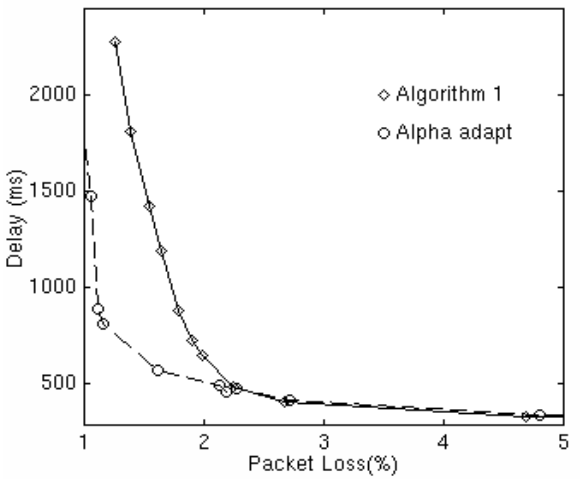

Fig. 6. Delay-loss comparison of α-adaptive algorithm with Algorithm 1.(Trace 3)

good as Algorithm 1.

B. Experiments based on the Jitter Control Procedure

The jitter control procedure has been applied to Algorithm 1 and tested on various traces for different values of *tolerance* allowed in silence zone compression. The results for varying *tolerance* are presented in Table II, on trace 4. The results there may be compared with those of Algorithm 1: delay 448.8ms and percentage loss=0.82%. It can be seen that for *tolerance* of 50%, the increase in delay is not significant and the procedure can be safely applied to improve the voice quality. Performance for this *tolerance* on different traces is presented in Table III.

C. Experiments on NLMS and Modifications

The NLMS algorithm has been implemented on traces from [5] with *ted* adjustment on a per talkspurt basis rather than per packet basis. We found in our simulations that the parameters μ and N which worked well for traces used in [4] did not work well for the traces from [5] with talkspurts, rendering the delay estimates useless. Hence, adjustments are made to these parameters. From the covariance lags plotted using (6), it was observed that the covariances became insignificant after about 10 past samples for trace 4, and after about 25 samples for trace 2.

Using $\mu = 0.0001$ and filter order, $N = 18$, the delay loss performance obtained is plotted in Figure 8 for one of the traces. In practice, to estimate N, the whole trace will not be available at the receiver, the covariance lags could be measured from the arrival times of the first few packets and used to decide the order of the filter. The performance of NLMS is not better than that of Algorithm 1 on either of the traces with talkspurt based adjustment.

For the reasons mentioned before, DWT has been applied to decorrelate the data before running it through NLMS. Beylkin, Daubechies and Haar wavelets were experimented with. The

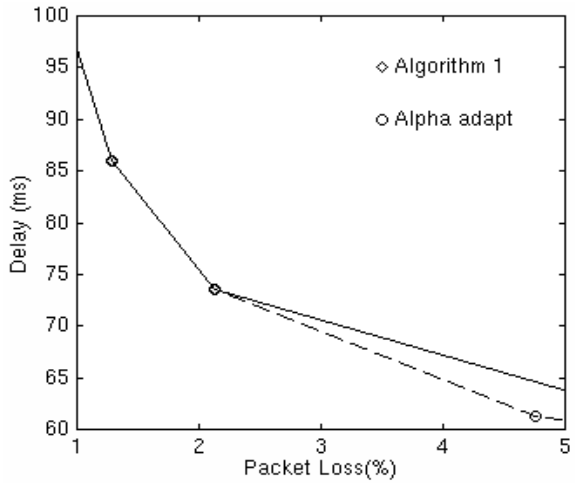

Fig. 7. Delay-loss comparison of α-adaptive algorithm with Algorithm 1.(Trace 4)

TABLE II
PROPOSED JITTER CONTROL PROCEDURE ON TRACE 2

Tolerance(%)	Loss(%)	Delay(ms)
50	0.82	449.6
75	0.82	450.5

choice of wavelet does not significantly affect the decorrelation process and the simple Haar wavelet can thus be used. The results for Haar wavelet based DWT are shown in Figure 8. It can be seen that DWT does improve the performance of NLMS and even makes it better than Algorithm 1 for some loss percentages.

VI. CONCLUSIONS

This paper has explored the main issues in buffering delay adjustment at the receiver for jitter free audio playout. A new algorithm has been presented to reduce the delays further at a given loss, based on actively adjusting the weighting parameter α in the AR delay estimete and it has been found to work better than conventional algorithms, especially for low packet loss percentages. A procedure to control jitter in the silence zone has also been described. Reduction in jitter using this procedure did not lead to serious increases in the playout delay. Hence this procedure may be safely incorporated into either of the playout algorithms. The performance of NLMS based playout algorithm has been studied. Our simulations illustrate that the algorithm as described in [4] does not work well for talkspurt based delay adjustment and we have suggested a modification based on Discrete Wavelet Transform which improves the performance of the algorithm. These suggestions can be used to improve the system performance for packet network based interactive audio applications. Further research is required to evaluate the perceptual quality of the played out voice, for each of the above modifications. In particular, the improvement offered by the new jitter control procedure needs

TABLE III
DELAY LOSS RESULTS FOR THE PROPOSED JITTER CONTROL PROCEDURE ON DIFFERENT TRACES

Trace	New Loss	New *ted*	Old Loss	Old *ted*
1	1.19%	924ms	1.19%	915ms
2	0.82%	449.6ms	0.82%	448ms
3	1.96%	606.1ms	1.97%	597.7ms
4	1.01%	91.72ms	1.01%	91.72ms

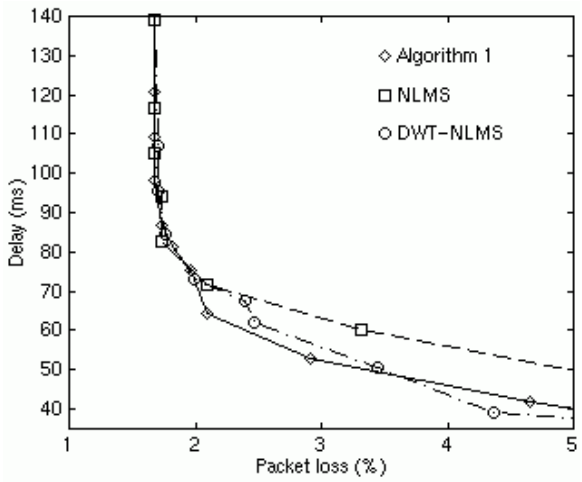

Fig. 8. Delay-loss comparison of NLMS based algorithm, NLMS with DWT and Algorithm 1.(Trace 4)

to be verified through Mean Opinion Score (MOS) tests.

REFERENCES

[1] X Wang and H Schulzrinne, "Comparison of adaptive internet multimedia applications," *IEICE Trans. Communications*, vol. E82-B, no. 6, pp. 806–818, June 1999.

[2] NS Jayant, "Effects of packet loss on waveform coded speech," in *Proc. Fifth Int. Conference on Computer Communications*, Atlanta, GA, October 1980, pp. 275–280.

[3] R Ramjee, Jim Kurose, Don Towsley, and Henning Schulzerine, "Adaptive playout mechanisms for packetized audio applications in wide area networks," *Proc. IEEE INFOCOMM*, pp. 680–686, June 1994.

[4] Phillip DeLeon and Cormac J Sreenan, "An adaptive predictor for media playout buffering," in *Proc. IEEE ICASSP*, Phoenix, Az, March 1999, pp. 3097–3100.

[5] Sue B Moon, Jim Kurose, and Don Towsley, "Packet audio playout delay adjustment: Performance bounds and algorithms," *Multimedia Systems*, vol. 6, pp. 17–28, January 1998.

[6] CJ Sreenan, Jyh-Cheng Chen, Prathima Agrawal, and B Narendran, "Delay reduction techniques for playout buffering," *IEEE Trans. Multimedia*, vol. 2, no. 2, pp. 88–100, June 2000.

[7] Jesus Pinto and Kenneth J Christensen, "An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods," in *Proc. 24th Conference on Local Computer Networks*, Lowell, Massachusetts, 7 - 20 October 1999, pp. 224–231.

[8] Harri Marjamaki and Raimo Kantola, "Performance evaluation of an ip voice terminal," in *Proc. Fifth IFIP Conference on Intelligence in Networks*, Asian Institute of Technology, Thailand, November 1999.