Proc. IEEE Canadian Conf. Electrical, Computer Engineering (Montreal, QC), pp. 997-1000, May 2003

AN ADAPTIVE PLAYOUT ALGORITHM WITH DELAY SPIKE DETECTION FOR REAL-TIME VOIP

Aziz Shallwani Peter Kabal

Department of Electrical and Computer Engineering
McGill University, Montréal, Canada
{ashallwani, kabal}@tsp.ece.mcgill.ca

Abstract

As the Internet is a best-effort delivery network, audio packets may be delayed or lost en route to the receiver due to network congestion. To compensate for the variation in network delay, audio applications buffer received packets before playing them out. Basic algorithms adjust the packet playout time during periods of silence such that all packets within a talkspurt are equally delayed. Another approach is to scale individual voice packets using dynamic time-scale modification.

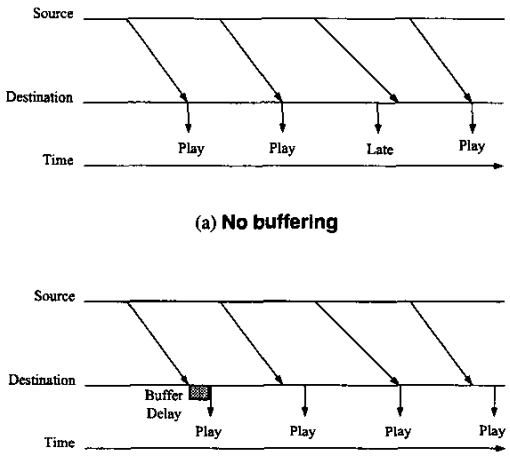
In this work, an adaptive playout algorithm based on the normalized least mean square algorithm, is improved by introducing a spike-detection mode to rapidly adjust to delay spikes. Simulations on Internet traces show that the enhanced bi-modal playout algorithm improves performance by reducing both the average delay and the loss rate as compared to the original algorithm.

Keywords: *adaptive playout buffer, jitter, VoIP, NLMS predictor*

1. Introduction

Voice transmission over the Internet is subject to network delay and loss. At the transmitter, speech/audio packets are generated at regular intervals and sent to the receiver. Ideally, the receiver would play the packets out at the same schedule. However, the network delay experienced by different packets may vary due to network congestion. The variation in network delay is referred to as *jitter*. If packets are played out at the receiver immediately upon arrival, there will be gaps in the playout because packets may arrive after their scheduled playout time, as illustrated in Fig. 1(a). The destination can reduce the "loss" due to late packets by storing received packets in a playout buffer before playing them out, as shown in Fig. 1(b).

Network delay traces are characterized by frequent occurrences of spikes in the network delays. Since end-to-end delays beyond 300 ms are irritating to users and impair interactivity in real-time conversations, the playout buffering delay is used to adjust the tradeoff between total end-to-end delay and loss rate. Packets with network delays greater than the playout delay will still be "lost", however 5% loss can be tolerated when packet loss concealment methods are applied. Adaptive playout buffer algorithms attempt to adjust the playout delay for changing network conditions.



(a) **No buffering**

(b) **Packets are buffered**

Fig. 1 The playout buffering problem [1]

Existing adaptive playout buffer algorithms are reviewed in the next section. Section 3 proposes the addition of a spike mode to the adaptive NLMS playout algorithm. The proposed algorithm is evaluated using delay traces and simulation results in Section 4 show a reduction in end-to-end delay and loss for the enhanced bi-modal NLMS algorithm.

2. Playout Buffer Algorithms

Adaptive playout buffer algorithms react to changing network conditions by dynamically adjusting the end-to-end delay. Since audio packets are generated at regular intervals, the received packets must be played out in a periodic manner. Playout delay adjustments made during periods of silence are less likely to be perceived by users. Therefore, the playout delay is adjusted on a per-talkspurt basis

CCECE 2003 - CCGEI 2003, Montréal, May/mai 2003
0-7803-7781-8/03/$17.00 © 2003 IEEE

by lengthening or shortening the silence between talkspurts.

The basic playout approach [2] is to set the playout time p_1^k of the first packet of talkspurt k to

$$p_1^k = t_1^k + D^k \quad (1)$$

where t_1^k and p_1^k are the sender timestamp and playout time respectively, of the first packet in talkspurt k and D^k is the total end-to-end delay for received packets in talkspurt k.

Subsequent packets in a talkspurt have the same total end-to-end delay. The playout time for packet i in talkspurt k can be calculated as an offset from the playout time of the first packet in the talkspurt, namely

$$p_i^k = p_1^k + (t_i^k - t_1^k) \quad (2)$$

In a recent approach to adaptive playout, playout delay adjustment is performed within talkspurts [3]. Individual voice packets are scaled such that they are played out just in time for the predicted arrival time of the next packet. A time-scale modification technique, based on the Waveform Similarity Overlap-Add (WSOLA) algorithm, is used to modify the playout rate while preserving the voice pitch. The degradation in perceptual quality due to scaling was found to be inaudible [3]. Dynamically adjusting the playout time improves overall performance by reducing end-to-end delay while keeping packet loss tolerable. The main approaches to playout delay estimation are described here.

2.1 Autoregressive (AR) Estimate-Based Algorithms

The basic playout algorithm uses an autoregressive (AR) estimate to compute the network delay and jitter [2]. The estimates for the average network delay, $\hat{d}_i$ and variation in network delay, $\hat{v}_i$ are given by

$$\hat{d}_i = \alpha \hat{d}_{i-1} + (1-\alpha) n_i \quad (3)$$

$$\hat{v}_i = \alpha \hat{v}_{i-1} + (1-\alpha)|\hat{d}_i - n_i| \quad (4)$$

where $\hat{d}_i$ is the autoregressive estimate of the packet delays, $\hat{v}_i$ is the variation in network delay, n_i is the network delay incurred by the i-th packet and α is a weighting factor used to control the adaptation rate of the algorithm.

The total end-to-end delay, D_i is computed as

$$D_i = \hat{d}_i + \beta \hat{v}_i \quad (5)$$

where β is a safety factor used to moderate the tradeoff between end-to-end delay and packet "loss" rate. A higher value of β results in a lower loss rate as more packets arrive in time, however the total end-to-end delay increases. 0.998002 and 4.0 are suggested for α and β, respectively [2]. While the AR estimates of $\hat{d}_i$ and $\hat{v}_i$ are updated for each packet, D^k is only set to D_i at the beginning of a new talkspurt.

A second algorithm uses two different values for α, one for increasing network delays and the other for decreasing network delays [2]. Minor changes in the value of α can greatly impact the tradeoff between packet loss and total end-to-end delay. The value of α determines how rapidly the AR delay estimate adapts to fluctuations in network delay. A modification to the basic algorithm adaptively adjusts α to an optimal value for the specified loss rate [4].

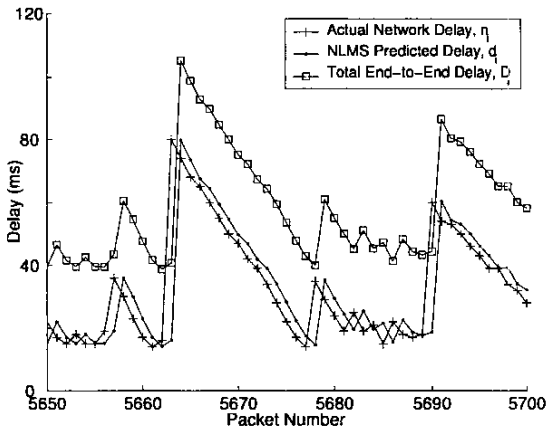


Fig. 2 NLMS Algorithm

2.2 Statistically-Based Algorithms

Statistically-based approaches use the statistics of past delays to compute the current playout delay. Network delays for the past w packets are stored and a playout delay is selected such that a chosen percentage of packets would have arrived in time [3, 5]. Only the delays of the past talkspurt are used in [6]. Another approach plots all previous delay values in a histogram and applies an aging procedure to gradually reduce the impact of older packet delays [7].

2.3 Adaptive Filter-Based Algorithms

Instead of reacting to network fluctuations, a novel approach to adaptive playout aims to predict the network delay [1]. An accurate prediction of the network delay can rapidly track network changes and thus adjust the playout delay more effectively, as shown in Fig. 2.

The basic adaptive filtering algorithm aims to minimize the expected mean square error between the actual data and the estimate [8]. Previously received data is passed through a finite-impulse response (FIR) filter to compute the current estimate. The mean square error is then used to adjust the tap weights of the adaptive filter.

The normalized least mean square (NLMS) algorithm is used for the adaptive predictor [1]. The estimate for the network delay, $\hat{d}_i$ is computed to be

$$\hat{d}_i = \mathbf{w}_i^T \mathbf{n}_i \quad (6)$$

where $\hat{d}_i$ is the predicted network delay value for packet i, $\mathbf{w}_i$ is the $N \times 1$ vector of adaptive filter coefficients, $()^T$ is the vector transpose, and $\mathbf{n}_i$ is the $N \times 1$ vector containing the past N network delays (up to and including the delay for packet $i-1$).

The filter tap weights, $\mathbf{w}_i$, are then updated after each packet using the NLMS algorithm [8]

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \frac{\mu}{\mathbf{n}_i^T \mathbf{n}_i + a}\mathbf{n}_i e_i \qquad (7)$$

where μ is the step size, a is a small constant to prevent division by zero, and $e_i = \hat{d}_i - n_i$ is the estimation error.

The network delay variation and total end-to-end delay are calculated as before using Eqs. (4) and (5). The total end-to-end delay is updated on a per-packet basis.

2.4 Spike Detection

The main playout buffer algorithms are not robust enough to adapt delay estimates in the presence of spikes. A spike is characterized by the sudden onset of a large increase in network delay. Although subsequent packets usually experience declining network delays, the delay values are still quite large. The spike ends when network delays return to average values. Fig. 2 depicts a typical delay spike.

A spike-detection algorithm was first developed by Ramjee et al. [2] to adapt to such spikes. The playout algorithm switches to an impulse or spike mode when the delay values of the previous two packets differ by more than a threshold. Within the spike, the network delays decline from the peak spike value and the delay estimate depends only on the most recent delay values. The slope of the delay spike is monitored and as the delays flatten out, the slope reduces and falls below a threshold, indicating the end of the spike. The algorithm then reverts to normal mode and the delay estimate is computed using the AR estimate-based approach. Other playout buffer algorithms also modify their delay estimates during spikes [3, 5, 6].

3. Enhanced NLMS (E-NLMS) Algorithm

A drawback of the NLMS predictor [1] is that it does not detect delay spikes and therefore does not alter its delay prediction during a spike. Fig. 2 illustrates the behaviour of the NLMS algorithm in the presence of a delay spike. The first packet in a delay spike arrives too late to be played out. The NLMS predictor will react and subsequent predictions will overestimate the ensuing delays. Thus the safety factor, β used to compute the playout delay can be significantly reduced for ensuing packets.

The E-NLMS algorithm takes advantage of this situation by adding a spike-detection mode to the NLMS predictor. In the normal mode of operation, the adaptive NLMS predictor functions as before [1]. A spike is detected when either the previous packet was lost or the actual delay value exceeds the predicted value by a threshold. Within the spike mode, the playout delay is still based on the NLMS delay prediction. Since the NLMS algorithm overestimates the packet delay for packets immediately following a spike, the value of the safety factor, β is reduced in computing the total end-to-end delay, D_i. However, the D_i in spike mode is not allowed to fall below its AR estimate. The spike ends when the NLMS delay prediction no longer exceeds the actual delay. The D_i is updated on a per-packet basis. Fig. 3 illustrates the behaviour of the enhanced bi-modal NLMS in the presence of a delay spike. The pseudocode of the algorithm is given below.

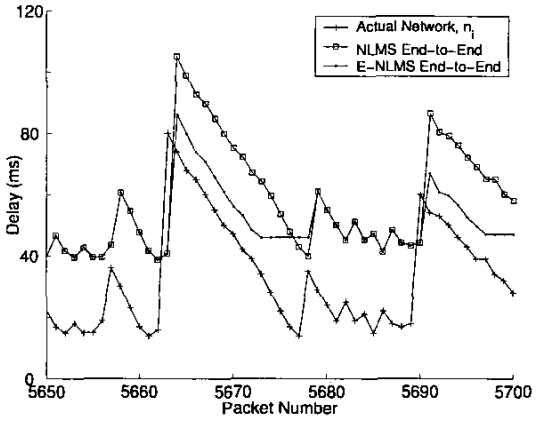


Fig. 3 Enhanced NLMS (E-NLMS) Algorithm

```
// Enhanced NLMS (E-NLMS) algorithm

d_i = w_i^T n_i;
w_{i+1} = w_i + μ/(n_i^T n_i + a) n_i e_i;

if ( mode == SPIKE )
    ARdelay_i = α ARdelay_{i-1} + (1 − α) n_{i-1};
    v_i = α v_{i-1} + (1 − α)|d_{i-1} − n_{i-1}|;
    varfactor_i = β/4  v_i;
    D_i = d_i + varfactor_i;
    if (D_i < ARdelay_i + β v_i)
       D_i = ARdelay_i + β v_i;
    end
else // Normal mode
    ARdelay_i = α ARdelay_{i-1} + (1 − α) n_{i-1};
    v_i = α v_{i-1} + (1 − α)|d_{i-1} − n_{i-1}|;
    varfactor_i = β v_i;
    D_i = d_i + varfactor_i;
end

// if end-to-end delay < network delay
if (D_i < n_i)
    packet_i=LOST;
else
    packet_i=IN_TIME;
end

if (n_i > d_i)
    mode = NORMAL;
end
```

```
if( (n_i > d_i + 5v_i) OR (packet_i==LOST) )
    mode = SPIKE;
end
```

4. Evaluation and Results

The E-NLMS algorithm was evaluated in comparison to the basic NLMS predictor for a set of delay traces. The basic *ping* program was modified to continuously send a 40 byte ICMP packet every 10 ms. A set of 18 traces, each trace lasting for 1 000 000 packets, was collected between nodes in North America. The traces were collected both during the day and at night. While paths within the Internet may not be symmetric, the round-trip delay gives a good idea of the magnitude of the actual network delay and avoids the problems of clock synchronization and clock skew. The E-NLMS and NLMS predictors were also tested on traces of one-way delays from [1].

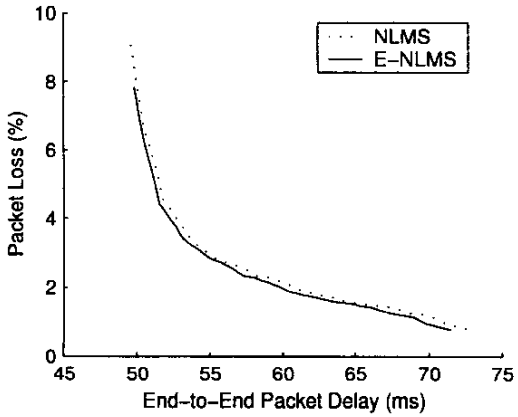


Fig. 4 Trace 1: NLMS and E-NLMS Alg.

The experiments measured the average total end-to-end delay and the late packet percentage for the delay traces. The tradeoff between the average end-to-end delay and the loss rate was illustrated by varying β. The results are shown in Figs. 4 and 5 for two sample traces. Trace 1 was taken between Montreal and Ottawa, while Trace 2 corresponds to trace "c1" from [1]. Reducing the delay overestimation during spikes leads to an improvment in overall performance. Both the average end-to-end delay and loss are reduced for the proposed E-NLMS algorithm as compared to the original NLMS predictor.

5. Conclusion

In this paper, an overview of the playout buffering problem has been presented. Moreover, the main adaptive playout buffering algorithms were reviewed. An existing NLMS predictor was enhanced by adding a spike mode to rapidly adjust to delay spikes. Within the spike mode, the enhanced NLMS algorithm makes use of the overestimate in the NLMS prediction and thus reduces the safety margin when computing the total end-to-end delay. Simulations on Internet traces compared the proposed E-NLMS algorithm to the original NLMS playout algorithm. The results demonstrate that the bi-modal algorithm improves the overall performance by reducing both the average end-to-end delay and the loss rate. The E-NLMS algorithm is well-suited for playout algorithms adjusting the delay on a per-packet basis.

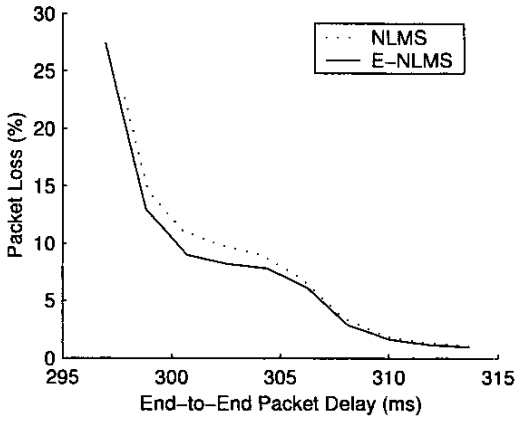


Fig. 5 Trace 2: NLMS and E-NLMS Alg.

References

[1] P. DeLeon and C. Sreenan, "An adaptive predictor for media playout buffering," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing* (Phoenix, AZ), pp. 3097–3100, Mar. 1999.

[2] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," *Proc. IEEE Conf. Comp. Commun. (IEEE-Infocom)* (Toronto, ON), pp. 680–688, June 1994.

[3] Y. J. Liang, N. Farber, and B. Girod, "Adaptive playout scheduling and loss concealment for voice communications over IP networks," *IEEE Trans. Multimedia*, to appear.

[4] A. Kansal and A. Karandikar, "Adaptive delay estimation for low jitter audio over internet," *Proc. IEEE Global Telecommun. Conf.* (San Antonio, TX), vol. 4, pp. 2591–2595, Nov. 2001.

[5] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: Performance bounds and algorithms," *ACM/Springer Multimedia Systems*, vol. 6, pp. 17–28, Jan. 1998.

[6] J. Pinto and K. J. Christensen, "An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods," *Proc. IEEE Conf. Local Computer Networks* (Lowell, MA), pp. 224–231, Oct. 1999.

[7] P. Agrawal, J.-C. Chen, and C. J. Sreenan, "Use of statistical methods to reduce delays for media playback buffering," *Proc. IEEE Int. Conf. Multimedia Computing and Systems* (Austin, TX), pp. 259–263, June 1998.

[8] S. Haykin, *Adaptive Filter Theory*. Upper Saddle River, NJ: Prentice Hall, third ed., 1996.