ON NLMS ESTIMATION FOR VOIP PLAYOUT DELAY ALGORITHMS
Improving delay spike detection

Karen S. Miranda-Campos and Víctor M. Ramos R.
Metropolitan Autonomous University, Electrical Engineering Department,
Networks & Telecommunications Research Team, Mexico
09340 Iztapalapa, Mexico
karen.miranda@ieee.org, vicman@xanum.uam.mx

Keywords: Playout control delay algorithms, VoIP, autoregressive estimation.

Abstract: Voice over IP (VoIP) applications are now very popular and widely used on the Internet. Such applications use receiver playout buffers to smooth delay variations so as to reconstruct the periodic form of the transmitted packets. Packets arriving after their scheduled playout time are considered late and are not played out. Playout delay control algorithms often operate by updating the playout delay between periods of silence. A recent class of playout control algorithms has received particular attention; this class of algorithms uses autoregressive measures on the network delay so as to estimate future packet delay values and adjust the playout delay accordingly. In this work, we compare two algorithms previously proposed that use such autoregressive approach; both playout algorithms use a normalized least-mean square (NLMS) adaptive predictor. The difference between both algoritms is that the second one is an extension of the first that adds delay spike detection. We demonstrate, by using Internet audio packet traces that, contrary on what was claimed, the algorithm that uses spike detection does not overperfom the first one. Finally, we propose an algorithm based on the original NLMS algorithm with delay spike detection that overperforms the previous two NLMS playout algorithms.

1 INTRODUCTION

Delay, jitter, and packet loss in packet-switched wide-area networks, such as the Internet, are the main factors impacting audio quality of interactive multimedia applications. Today's Internet still operates in a best-effort basis, and thus, the impact caused by such phenomena must be alleviated by employing end-to-end control mechanisms. Audio applications such as NeVoT (Schulzrinne, 1992), Rat (Sasse and Hardman, uary), or more recently Skype and Google Talk generate packets spaced at regular time intervals. The traffic generated by an audio source is divided into periods of activity, called *talkspurts*, and periods of silence. Silence periods are periods where no audio packets are transmitted.

Audio packets encounter variable delay while crossing the Internet, this is mainly due to the variable queueing time in routers. Such delay variability modifies the periodic form of the transmitted audio stream. In order to playout the received stream, an application must reduce or eliminate this delay variability, by buffering the received packets and playing them out after a certain deadline. Packets arriving after their corresponding deadline are considered late and are not played out. If the playout delay is increased, the probability that a packet will arrive before its scheduled playout time also increases. This reduces the number of packets artificially dropped in the playout buffer. However, very long playout delays have a negative impact on the interactivity of an audio session. Obviously, there exists a trade-off between delay and loss due to late packets. For interactive audio, packet delays up to 400 ms and loss rates up to 5%–10%, depending on the audio codec used, are considered adequate (Jayant, 1980).

In this work, we compare two algorithms previously proposed in the literature that use an autoregressive approach; both playout algorithms use a normalized least-mean square (NLMS) adaptive predictor. The difference between both algoritms is that the second one is an extension of the first that adds delay spike detection. From here, we call this second algorithm E-NLMS. We demonstrate, by using real In-

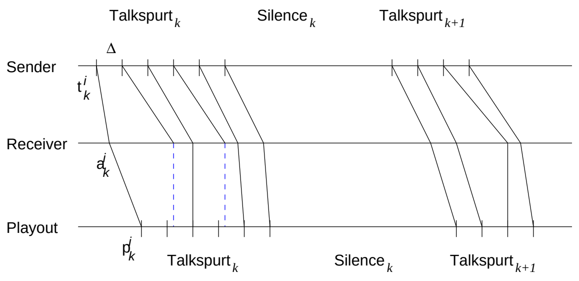


Figure 1: The timings between the transmission, reception and playout of packets.

ternet audio packet traces that, contrary on what was claimed, the E-NLMS algorithm does not overperfom the original NLMS algorithm. Finally, we propose an improvement with spike detection that overperforms both, the NLMS and the E-NLMS algorithms.

The remainder of the paper is as follows. In Section 2, we provide some background about playout delay algorithms. In Section 3 we describe the related work on which we base our proposal, and particularly, on the algorithms using the NLMS playout algorithms just cited. In Section 4, we present our ongoing work on an improvement of the original NLMS algorithm proposed by DeLeon. Finally in Section 5, we conclude and present our current and future work.

2 BACKGROUND

Receivers use a playout buffer to smooth the stream of audio packets. This smoothing is done by delaying the playout of packets to compensate for variable network delay. The playout delay can be either constant through the whole session, or can be adjusted between talkspurts. Moreover, in a recent work (Liang et al., 2001), it has been shown that by using a technique called *packet scaling*, it is possible to change the playout delay from packet to packet while keeping the resulting distortion within tolerable levels. In this paper we only focus on the per-talkspurt playout delay adjusting approach.

Figure 1 shows the different stages incurred in an audio session. The i-th packet of talkspurt k is sent at time t_k^i, it arrives at the receiver at time a_k^i, and is held in the smoothing receiver's playout buffer until time p_k^i, when it is played out. Within a talkspurt, packets are equally spaced at the sender by time intervals of length Δ seconds.

By delaying the playout of packets and dropping those that arrive after their deadline, we are able to reconstruct the original periodic form of the stream. This adjusting mechanism results in a regenerated stream having stretched or compressed silence periods compared to the original stream. These changes are not noticeable by the human ear if they are kept within tolerable small levels.

In Fig. 1, a dropped packet due to a late arrival is represented by a dashed line. A packet is artificially dropped if it arrives after its scheduled deadline p_k^i. The loss percentage can be reduced by increasing the amount of time that packets stay in the playout buffer. An efficient playout control algorithm must take into account the trade-off between loss and delay in order to keep both parameters as low as possible.

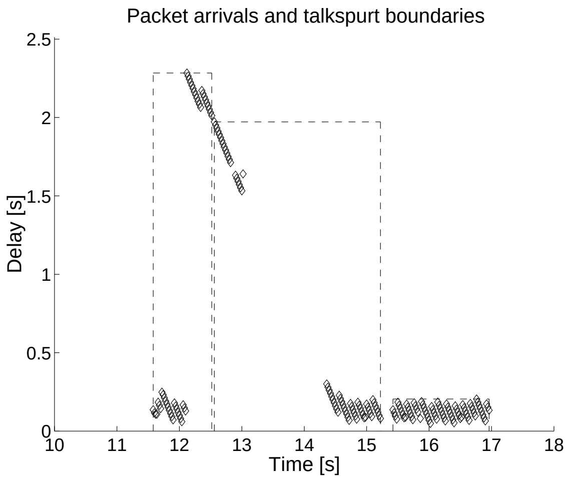
Throughout the paper, we use the notation described in Table 1. For the validation of our algorithm, we consider the packet traces generated with the NeVoT audio tool that are described in (Moon et al., 1998). We choose to use those traces since they are packet traces generated during real audio conversations. In (DeLeon and Sreenan, 1999) and later in (Shallwani and Kabal, 2003), DeLeon and Shallwani use traces generated with the *ping* program for the latter, and traces generated between three hosts in the US and one host in the UK. Those traces are not available to everyone and thus we cannot verify the delay behavior in them. On the other side, the audio traces provided by Moon, contain the sender and receiver timestamps of transmitted packets that are needed for the implementation of any playout delay control algorithm. In these traces, one 160 byte audio packet is generated approximately every 20 ms when there is speech activity. A description of the traces (reproduced from (Moon et al., 1998)) is depicted in Table 2.

A typical sample of packet end-to-end delays is shown in Fig. 2. A packet is represented by a diamond and talkspurt boundaries by dashed rectangles. The x-axis represents the time elapsed at the receiver since the beginning of the audio session. Only the variable portion of the end-to-end delay (d_k^i) is represented on the y-axis of Fig. 2. To this end, the constant component of the end-to-end delay (mostly caused by the propagation delay) is removed by subtracting from packet delays their minimum over all the corresponding trace. By considering the variable portion of the end-to-end delay, synchronization between sender and receiver clocks can be avoided.

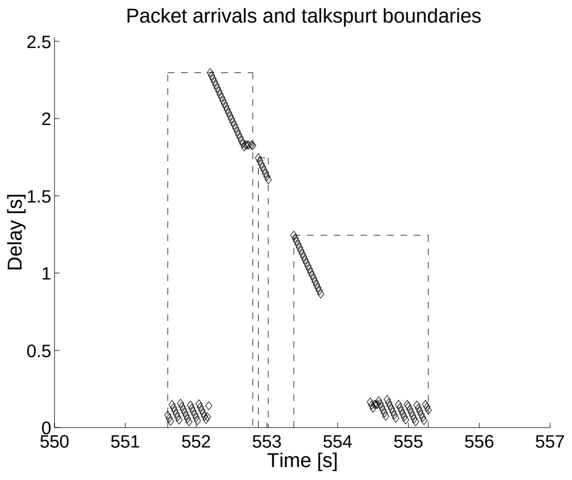
We observe in Fig. 2 the presence of delay spikes. This phenomenon in end-to-end delay has been previously reported in the literature (Ramjee et al., 1994; Bolot, 1993). Delay spikes represent a serious problem for audio applications since they affect the performance of playout delay adaptation algorithms. A *delay spike* is defined as a sudden large increase in the end-to-end delay followed by a series of packets arriv-

Table 1: Definition of variables.

Param.	Meaning
L	The total number of packets arriving at the receiver during a session.
N	The total number of talkspurts in a session.
N_k	The number of packets in talkspurt k.
t_k^i	The time at which the i-th packet of talkspurt k is generated at the sender.
a_k^i	The time at which the i-th packet of talkspurt k is received.
d_k^i	The variable portion of the end-to-end delay of the i-th packet in talkspurt k. $d_k^i = a_k^i - t_k^i - \min_{\substack{1\le k\le N \\ 1\le i\le N_k}} (a_k^i - t_k^i)$.
p_k^i	The time at which packet i of talkspurt k is played out.



(a) A delay spike spanning through two consecutive talkspurts.



(b) A delay spike spanning through three consecutive talkspurts.

Figure 2: Delay spikes in end-to-end delay measurements.

ing almost simultaneously, leading to the completion of the spike (Ramjee et al., 1994).

Delay spikes can be contained within a single talkspurt or can span over several talkspurts. Figure 2(a) shows a delay spike spanning through two consecutive talkspurts. Figure 2(b) shows a delay spike spanning over three talkspurts. Since the playout delay is generally updated between talkspurts, a playout algorithm behaves better when delay spikes span over more than one talkspurt. Only in this way, a playout algorithm can react adequately to the spike by setting the playout delay according to the experienced delay. If the spike vanishes before the end of a talkspurt, the playout algorithm will not have enough time to set the playout time accordingly.

In the next section, we briefly describe the algorithms proposed by DeLeon. Playout delay is adapted from talkspurt to talkspurt based on past statistics of the delay process. The playout delay of the first packet of each talkspurt is the basetime of the deadlines for subsequent packets in the same talkspurt. This principle is the basis for most of the existing playout adaptation algorithms (Kansal and Karandikar, 2001; Ramjee et al., 1994; Moon et al., 1998; Pinto and Christensen, 1999; Ramos et al., 2003).

3 Related Work

The algorithms that perform better in (Ramjee et al., 1994) are 1 and 4. Algorithm 1 estimates the delay using a FIR filter; algorithm 4 operates similarly but it adds delay spike detection. To calculate $\hat{d}_k^i$ and $\hat{v}_k^i$, the packet's sender and receiver timestamps, t_k^i and a_k^i, are read from a trace file. Both algorithms differ only in the way they calculate $\hat{d}_k^i$ and $\hat{v}_k^i$. Algorithm 1 computes these statistics as follows:

$$\hat{d}_k^i = \alpha \hat{d}_k^{i-1} + (1-\alpha) d_k^i$$

Table 2: Description of the traces.

Trace	Sender	Receiver	Start time	Length [s]	Talkspurts	Packets
1	UMass	GMD Fokus	08:41pm 6/27/95	1348	818	56979
2	UMAss	GMD Fokus	09:58am 7/21/95	1323	406	24490
3	UMAss	GMD Fokus	11:05am 7/21/95	1040	536	37640
4	INRIA	UMass	09:20pm 8/26/93	580	252	27814
5	UCI	INRIA	09:00pm 9/18/93	1091	540	52836
6	UMass	Osaka University	00:35am 9/24/93	649	299	23293

and

$$\hat{v}_k^i = \alpha \hat{v}_k^{i-1} + (1-\alpha)|\hat{d}_k^i - d_k^i|,$$

where $d_k^i = a_k^i - t_k^i$, and α has the default value of 0.998002. Once $\hat{d}_k^i$ and $\hat{v}_k^i$ are computed, the playout time of the i-th packet of talkspurt k is set by both algorithms as follows:

$$p_k^i = \begin{cases} t_k^i + \hat{d}_k^i + \beta \hat{v}_k^i, & \text{for } i = 1 \ . \\ p_k^1 + (t_k^i - t_k^1), & \text{for } 1 < i \leq N_k \ . \end{cases} \quad (1)$$

This is how most algorithms falling into this class operate.

From now on, we focus only on the algorithms proposed first by DeLeon (NLMS) in (DeLeon and Sreenan, 1999) and later by Shallwani (E-NLMS) (Shallwani and Kabal, 2003). The work in (Shallwani and Kabal, 2003) aims to improve the NLMS algorithm proposed by DeLeon by introducing delay spike detection. Table 3 shows the parameters of the NLMS algorithm used in both papers:

The algorithm proposed by DeLeon uses a simple adaptive NLMS approach:

$$\mathbf{h}_{i+1} = \mathbf{h}_i + \frac{\mu}{\mathbf{n}_i^T \mathbf{n}_i + a} \mathbf{n}_i e_i \quad (2)$$

This NLMS algorithm minimizes the mean squared error (mse) between the current delay sample and its estimate. Previous samples are passed through a FIR filter to compute the current estimate. The mse is then used to update the weights of the adaptive filter.

Later, Shallwani proposed an extension of the DeLeon algorithm by adding delay spike detection which we call E-NLMS. He claims that, by adding spike detection the performance of the NLMS algorithm for playout delay is improved. We show in next section that the latter is not true for most of the cases using real audio traces rather than ping traces measuring the RTT.

Table 3: NLMS parameters.

Parameter	Value
h_0	$[1\ 0 \ldots 0]$
N_{nlms}	18
μ	0.01

4 Results

Before we can test the algorithms proposed by DeLeon (NLMS) and Shallwani (E-NLMS), and next the improvement we propose to improve the performance of the NLMS algorithm, we define the set of performance measures we use in our paper. To assess the performance of a playout adaptation algorithm, we focus on the total number of packets that are played out during an audio session, as well as on the experienced average end-to-end delay. Suppose we are given a packet audio trace with the sender and receiver timestamps of audio packets. Let p_k^i, N, L, N_k, t_k^i, and a_k^i be defined as in Table 1. As in (Moon et al., 1998), we define r_k^i to be a variable indicating if packet i of talkspurt k is played out or not. So, r_k^i is defined as:

$$r_k^i = \begin{cases} 0, & \text{if } p_k^i < a_k^i \ . \\ 1, & \text{otherwise.} \end{cases}$$

The total number of packets, T, played out in an audio session is thus given by:

$$T = \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i . \quad (3)$$

The average playout delay, D_{avg}, is equal to :

$$D_{avg} = \frac{1}{T} \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i [p_k^i - t_k^i]. \quad (4)$$

Finally, the loss percentage, l, is equal to :

$$l = \frac{L-T}{L} \times 100. \quad (5)$$

With the performance measures defined above, we choose to replace the delay spike detection in the

Shallwani's algorithm by a modification of the delay spike detection proposed by Ramjee. We choose to adjust the `var` parameter in line 8 of the Algorithm shown bellow to an adaptive value as a function of the previous recent delay spikes. The original NLMS algorithm (DeLeon) is kept during the "`NORMAL`" mode.

Algorithm 1 Algorithm 4 by Ramjee et al.

```
1:  $n_i = Receivertimestamp - Sendertimestamp;$
2:  if $(mode == NORMAL)$ then
3:      if$(abs(n_i - n_{i-1}) > abs(\hat{v}) * 2 + 800);$
4:      $var = 0;$
5:      $mode = IMPULSE;$
6:  else
7:      $var = var/2 + abs((2n_i - n_{i-1} - n_{i-2})/8);$
8:      if $(var <= 63);$ then
9:          $mode = NORMAL;$
10:         $n_{i-2} = n_{i-1};$
11:         $n_{i-1} = n_i;$
12:         return;
13:     end if
14: end if
15: if $mode == NORMAL$ then
16:     $\hat{d}_i = 0.125 * n_{i-1} = 0.875 * \hat{d}_{i-1};$
17: else
18:     $\hat{d}_i = \hat{d}_{i-1} = n_i = n_{i-1};$
19: end if
20: $\hat{v}_i = 0.125 * abs(n_i - \hat{d}_i) + 0.875 * \hat{d}_{i-1};$
21: $n_{i-2} = n_{i-1};$
22: $n_{i-1} = n_i;$
23: return;
```

We call our new NLMS algorithm **NLMS-mod**. Then we test our algorithm with the packet audio traces provided by Sue Moon and compare it with the DeLeon's and Shallwani's algorithms.

Figure 3 shows the corresponding results for traces 1 and 3. From the figures we note the following:

- Our NLMS-mod algorithm overperforms the two other algorithms. Moreover, it behaves better for the loss rates of interest (bellow 5%–10% depending on the audio codec used).
- One very important result is that, contrary on what was claimed by Shallwani, the E-NLMS algorithm he proposed in (Shallwani and Kabal, 2003) does not perform better than the DeLeon's NLMS algorithm. This is a a pretty important result, since it demonstrates that the delay process of ping packets may not model well a delay process obtained by a real voice conversation. Moreover, this result might reflect a bug in the Shallwani's algorithm.

5 CONCLUSIONS AND FUTURE WORK

We presented in this paper an improvement of the original NLMS algorithm for playout delay control proposed by DeLeon in (DeLeon and Sreenan, 1999). With trace-based simulation, we compared our algorithm with DeLeon's and Shallwani's algorithms and we proved that our NLMS-mod algorithm overperforms them for the loss rates of interest in interactive applications as VoIP.

An additional result is that we prove that, contrary on what was claimed, the Shallwani's E-NLMS algorithm does overperform the DeLeon's algorithm.

Our future work will focus on adapting the size of the memory of the NLMS algorithm in a dynamic fashion, so the memory size of the predictor adapts to the network state. Other research directions we are interested are exploring the behavior of different autoregressive estimators for playout delay control.

REFERENCES

Bolot, J. (1993). End-to-end packet delay and loss behavior in the Internet. In *Proceedings of the ACM SIGCOMM*, pages 289–298.

DeLeon, P. and Sreenan, C. (1999). An adaptive predictor for media playout buffering. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3097–3100.

Jayant, N. (1980). Effects of packet loss on waveform coded speech. In *Proceedings of the International Conference on Computer Communications*, pages 275–280.

Kansal, A. and Karandikar, A. (2001). Jitter-free audio playout over best effort packet networks. In *ATM Forum International Symposium*, New Delhi, India.

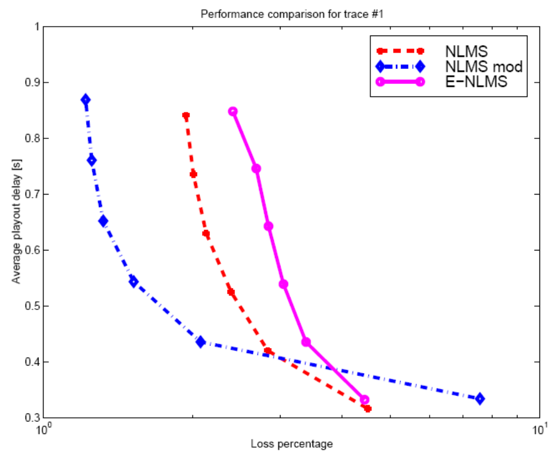
Liang, Y. J., Farber, N., and Girod, B. (2001). Adaptive playout scheduling and loss concealment for voice communications over IP networks. *IEEE Transactions on Multimedia*.

Moon, S. B., Kurose, J., and Towsley, D. (1998). Packet audio playout delay adjustment: Performance bounds and algorithms. *ACM/Springer Multimedia Systems*, 6:17–28.

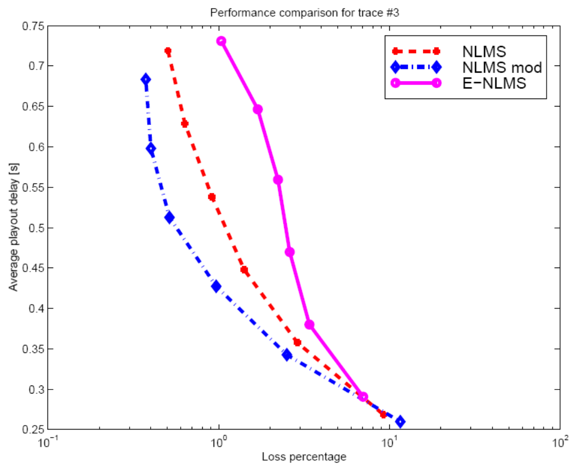
Pinto, J. and Christensen, K. J. (1999). An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods. In *Proceedings of the IEEE Conference on Local Computer Networks*, pages 224–231.

Ramjee, R., Kurose, J., Towsley, D., and Schulzrinne, H. (1994). Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the IEEE Infocom*, pages 680–688.

Ramos, V., Barakat, C., and Altman, E. (2003). A moving average predictor for playout delay control in VoIP.



(a) Comparing DeLeon's, Shallwani's and NLMS-mod algorithms with Trace 1.



(b) Comparing DeLeon's, Shallwani's and NLMS-mod algorithms with Trace 3.

Figure 3: Performance comparison.

In *Proceedings of the XI International Workshop on Quality of Service*.

Sasse, A. S. and Hardman, V. (February). Multi-way multicast speech for multimedia conferencing over heterogeneous shared packet networks. RAT-robust audio tool. Technical report, EPSRC Project #GRIK72780.

Schulzrinne, H. (1992). Voice communication across the Internet: a network voice terminal. Technical report, University of Massachusetts, Amherst.

Shallwani, A. and Kabal, P. (2003). An adaptive playout algorithm with delay spike detection for real-time VoIP. In *Proceedings of the IEEE Canadian Conference on Electrical Computer Engineering*, pages 997–1000.