AN ADAPTIVE PREDICTOR FOR MEDIA PLAYOUT BUFFERING

Phillip DeLeon

New Mexico State University
Klipsch School of Electrical and Computer Engineering
Las Cruces, NM 88003
pdeleon@nmsu.edu

Cormac J. Sreenan

AT&T Labs - Research
Florham Park, NJ 07932
cjs@research.att.com

ABSTRACT

Receiver playout buffers are required to smooth network delay variations for multimedia streams. Playout buffer algorithms such as those commonly used in the Internet, autoregressively measure the network delay and variation and adjust the buffer delay accordingly, to avoid packets arriving too late. In this work, we attempt to adjust the buffer delay based on a *prediction* of the network delay and a similar measure of variation. The philosophy here is that the use of an accurate prediction will adjust the buffer delay more effectively by tracking rapid fluctuations more accurately. Proper buffer delay can lead to either (or both) a lower total end-to-end delay for a fixed packet lateness percentage or fewer late packets for a fixed total end-to-end delay which are both important metrics for applications such as IP telephony. We present a playout algorithm based on a simple normalized least-mean-square (NLMS) adaptive predictor and demonstrate using Internet packet traces that it can yield reductions in average total end-to-end delays.

1. INTRODUCTION

The Internet and other packet networks are now used to transport audio and video streams, supporting applications such as conferencing and telephony. A characteristic of these networks is that the total delay experienced by each packet is a function of variable delays due to physical media access and queuing in routers and switches, in addition to fixed propagation delays. This variation in network delay, known as jitter, means that the time difference between transmitting any two packets at the sender is unlikely to be the same as that observed between their arrival at the receiver. Multimedia applications generate packets according to a schedule (e.g. every 40 milliseconds) and ideally the receiver should reproduce that schedule for accurate playout. Due to network jitter, if a receiver simply plays out packets as they arrive, there will be gaps in the playout because of packets which arrive later than their scheduled playout time as shown in Figure 1(a). To deal with network jitter, common practice is to use a per-stream receiver playout buffer, which attempts to smooth jitter prior to presenting the data to users as shown in Figure 1(b). A playout buffer operates by introducing an additional buffer delay and holding packets until their scheduled playout time. Naturally, if the playout buffer delay is too small then some packets will still arrive too late and be discarded, causing gaps. On the other hand, selecting a large buffer delay results in large end-to-end delays which are undesirable for interactive applications such as telephony.

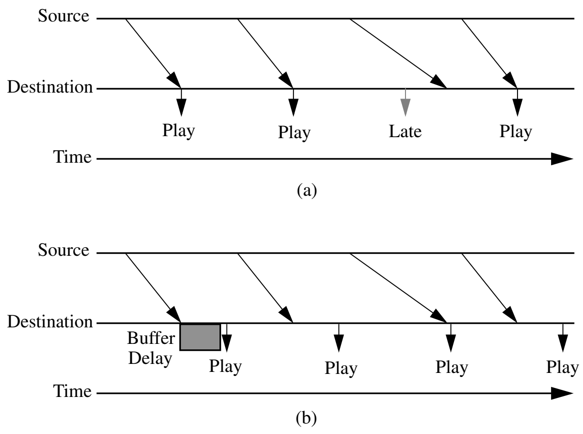

Figure 1: Playout problem: (a) packet arrives too late, (b) solution using a buffer delay.

Common algorithms for managing a playout buffer take a reactive approach, recognizing that network conditions and hence jitter do change over the lifetime of a stream. Instead of trying to determine *a priori* the most suitable buffer delay, these reactive algorithms compute an average network delay and use that in determining a current suitable buffer delay so that the percentage of late packets is kept low, typically well under 1%. The most common approaches use an autoregressive average of the network delay in the selection of the total end-to-end delay (ted). End-to-end delay for a packet is equal to the accumulated network delay plus buffer delay. As an example, [1] provides a concise description of the algorithm used in the NeVoT Internet audio tool. In this case the delay average upon arrival of packet i is calculated as

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

and the variation is calculated as

$$\hat{v}_i = \alpha \hat{v}_{i-1} + (1-\alpha) |\hat{d}_i - n_i| \tag{2}$$

Here $\hat{d}_i$ is the one-way network delay estimate, $\hat{v}_i$ is an estimate of the variation in network delay, n_i is the actual network delay for packet i, and α is a weighting factor that controls the rate of convergence of the algorithm. Given these values the ted is set using

$$ted = \hat{d}_i + \beta \hat{v}_i \tag{3}$$

where β is a factor chosen to accommodate changes in network conditions that occur suddenly. In [1], α was chosen to be 0.998002 and β was chosen to be 4.0. For voice streams using silence suppression it is common to adjust ted only at the start of a talk spurt, thereby minimizing the impact on audio quality of delay changes. In other situations ted can be adjusted periodically, or based on a threshold difference between target and current ted.

There have been several other algorithms for playout buffer operation. Most recently [2] proposed maintaining a statistical representation of network delays observed during the stream lifetime. This allows applications to explicitly trade off future end-to-end delay and late packets. The work of [3] allows a similar trade off based on a record of Internet network delays for a fixed number of packets, currently 10000. In [4] an algorithm is presented which monitors variations in receiver buffer lengths to determine playout delay, rather than trying to estimate network delays.

In this work, we adjust the buffer delay based on a *prediction* of the network delay and a measure of variation similar to that of the reactive approach. The philosophy here is that the use of an accurate prediction as opposed to an autoregressive average will adjust the buffer delay more effectively by tracking rapid fluctuations more accurately. Proper buffer delay can lead to either (or both) a lower total end-to-end delay for a fixed packet lateness percentage or fewer late packets for a fixed total end-to-end delay which are both important metrics for applications such as IP telephony. In this paper, we present a playout algorithm based on a simple normalized least-mean-square (NLMS) adaptive predictor and demonstrate using Internet packet traces that it can yield reductions in average total end-to-end delays.

The paper is organized as follows. Section 2 describes how we apply adaptive prediction in the design of a playout buffer algorithm. Section 3 presents the results of an experimental comparison between our algorithm and the reactive algorithm described above. Section 4 concludes.

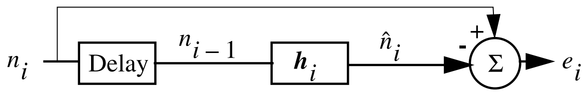

Figure 2: Adaptive one-step predictor.

Table 1: NLMS Algorithm fixed parameters.

Parameter	*Value*
$\mathbf{h}_0$	$[1\ 0\ \ldots\ 0]^T$
N	18
$\tilde{\mu}$	0.01

2. ALGORITHM

Adaptive filtering algorithms have been successfully applied in a number of areas such as equalization, echo cancellation, and prediction. In these applications, an adaptive algorithm seeks to minimize the expected squared error (cost function) between the actual data, n_i and the estimate, $\hat{n}_i$ by adjusting the coefficients of a finite-impulse response filter, $\mathbf{h}_i$ used in cascade or in parallel to the system. Figure 2 illustrates an adaptive one-step predictor.

One of the simplest adjustment algorithms is the NLMS algorithm given by

$$\mathbf{h}_{i+1} = \mathbf{h}_i + \frac{\tilde{\mu}}{\mathbf{n}_{i-1}{}^T \mathbf{n}_{i-1} + a} \mathbf{n}_{i-1} e_i \tag{4}$$

where $\mathbf{h}_i$ is the $N \times 1$ vector of adaptive filter coefficients, $\tilde{\mu}$ is the step-size, $\mathbf{n}_{i-1}$ is the $N \times 1$ vector containing the most recent N network delays, ^{T} is vector transpose, and e_i is the estimation error given by

$$e_i = n_i - \hat{n}_i = n_i - \mathbf{h}_i^T \mathbf{n}_{i-1} \tag{5}$$

where $\hat{n}$ is the estimate for packet i [5].

In the application of the NLMS adaptive filter to the selection of the ted, we perform the following for each packet:

1. compute the network delay prediction $\hat{n}_i$ given N previous network delays $n_{i-1}, \ldots, n_{i-N}$ using the NLMS algorithm

2. autoregressively calculate the variation as in the reactive algorithm (Eqn. 2)

3. choose the ted as in Eqn. 3 but using the network delay prediction, $\hat{n}_i$ instead of $\hat{d}_i$.

In our algorithm we use the parameters listed in Table 1.

We note that the algorithm requires little computation and memory and is attractive for real-time implementation.

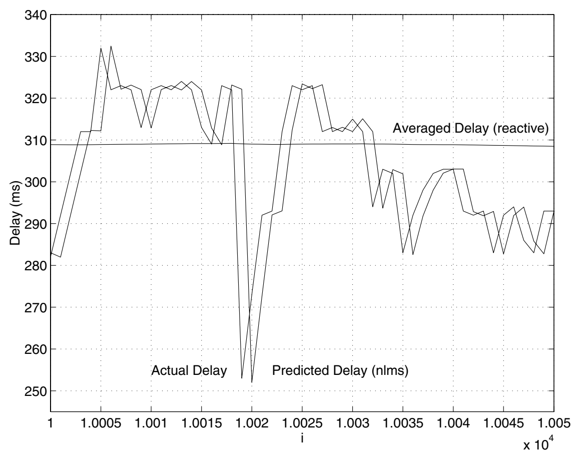

Figure 3: Comparing the reactive and NLMS algorithms.

3. RESULTS

To evaluate the adaptive prediction playout algorithm we compared it to the aforementioned reactive algorithm using a set of traffic traces. The traces were recorded for streams traversing the Internet to AT&T in New Jersey (`research.att.com`) from three hosts in the US, and one in the UK. Each trace lasted 10 minutes; traces were captured from each host at different times of the day. Each UDP packet carried 20 msec worth of audio and contained a sequence number and sender timestamp. We recorded the arrival time and used the differences to estimate network delay variations. The sender/receiver clocks were not synchronized, so these differences include any clock offset as well as network delay. Thus for our purposes we define network delay as that quantity in excess of the minimum observed delay from sender to receiver for a given stream. The latter is determined by finding the packet with the smallest difference in send/receive times (we assume negligible clock drift over the lifetime of each trace).

In the experiments that follow we measure the average ted in milliseconds, the percentage of actual late packets (alp) and the average number of packets between late packets ($dist$). These are collected across a set of 12 traces and for both the reactive and NLMS playout algorithms. Runs are listed according to their source, i.e. Stanford University (s), Carnegie-Mellon University (m), Rutgers University (r) and Cambridge University (c). Runs using the NLMS algorithm are shown in boldface. We report on the results of two experiments.

Figure 3 shows a time-sequence segment comparing the reactive and NLMS algorithms. Notice that the autoregressive approach of the reactive algorithm produces an average, whereas NLMS predicts the delay for each packet.

In experiment 1 we compare the two algorithms using published parameters for reactive on the sample traces measuring both average ted, alp and $dist$. We observe from Table 2 that while the NLMS algorithm has an increase in alp, there is a decrease in ted. We note however, that the alp for NLMS is still generally under 0.2% which is acceptable for the intended applications. Finally, we note that the late packets are on average spread very far apart which is also desirable for the intended applications.

In experiment 2 we adjust β for the reactive algorithm permitting more late packets than in experiment 1 while attempting to lower the average ted. We observer from Table 3 that when permitting late packets (but still attempting to stay under 0.5%) the NLMS algorithm in general maintains a lower average ted *and* a lower alp. This suggests that when we wish to lower the average ted at the expense of increasing the alp, we can do better by employing the predictive technique.

4. CONCLUSION

In this paper we have demonstrated the use of an adaptive network delay predictor in selecting the total end-to-end delay for smoothing network jitter using a playout buffer. Compared to conventional reactive techniques which use an autoregressive average of the network delay, this predictive technique in general can produce lower average total end-to-end delays and simultaneously a lower percentage of late packets when late packets are permitted as compared to reactive algorithms.

5. REFERENCES

[1] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pp. 680–688, June 1994.

[2] P. Agrawal, J.-C. Chen, and C. J. Sreenan, "Use of statistical methods to reduce delays for media playback buffering," in *Proceedings IEEE Multimedia Computing and Systems Conference (ICMCS)*, pp. 259–263, June 1998.

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

[4] D. L. Stone and K. Jeffay, "An empirical study of delay jitter management policies," *ACM/Springer-Verlag Multimedia Systems Journal*, vol. 2, pp. 267–279, Jan. 1995.

[5] S. Haykin, *Adaptive Filter Theory*. Englewood-Cliffs, N.J.: Prentice-Hall, 3 ed., 1998.

Table 2: Experiment 1: Reactive ($\alpha = 0.998002, \beta = 4.0$) and in boldface NLMS ($\alpha = 0.99, \beta = 6.0$).

Run	ted	alp	$dist$
c1	352.3	0.00	-
c1	**344.6**	**0.04**	**2322.1**
c2	435.5	0.00	-
c2	**418.2**	**0.06**	**1503.0**
c3	758.5	0.00	-
c3	**748.8**	**0.06**	**1368.0**
m1	2503.0	0.00	-
m1	**2474.4**	**0.10**	**908.6**
m2	287.9	0.00	-
m2	**278.3**	**0.14**	**714.6**
m3	2725.6	0.00	-
m3	**2702.0**	**0.08**	**1245.8**
r1	108.8	0.10	781.2
r1	**109.9**	**1.70**	**53.5**
r2	120.7	0.06	1108.6
r2	**122.0**	**1.60**	**61.2**
r3	2508.0	0.31	178.1
r3	**2475.1**	**1.59**	**59.4**
s1	211.3	0.00	-
s1	**197.1**	**0.07**	**1366.4**
s2	283.7	0.00	-
s2	**265.8**	**0.10**	**963.8**
s3	224.2	0.40	222.3
s3	**214.3**	**0.04**	**2837.8**

Table 3: Experiment 2: Reactive ($\alpha = 0.998002, \beta = 2.6$) and in boldface NLMS ($\alpha = 0.99, \beta = 4.0$).

Run	ted	alp	$dist$
c1	331.5	0.44	205.9
c1	**326.0**	**0.19**	**537.0**
c2	410.5	0.26	350.7
c2	**398.2**	**0.20**	**481.2**
c3	739.6	0.10	450.9
c3	**730.1**	**0.15**	**602.8**
m1	2482.1	0.00	-
m1	**2452.1**	**0.36**	**261.6**
m2	260.3	0.63	139.2
m2	**254.0**	**0.47**	**208.8**
m3	2709.8	0.00	-
m3	**2682.4**	**0.32**	**302.8**
r1	105.3	0.28	295.6
r1	**105.8**	**3.10**	**31.7**
r2	116.2	0.21	336.0
r2	**116.8**	**3.28**	**30.2**
r3	2503.5	0.56	109.3
r3	**2470.8**	**2.82**	**34.6**
s1	193.9	1.39	67.9
s1	**184.5**	**0.17**	**555.2**
s2	265.0	0.10	678.3
s2	**252.6**	**0.21**	**489.6**
s3	201.7	3.24	28.5
s3	**194.8**	**0.30**	**321.1**