Prediction schemes for playout delay control in VoIP

Víctor M. Ramos R. *†, Chadi Barakat ‡and Eitan Altman §

September 20, 2002

Abstract

Audio applications are now extensively used in packet-switched wide-area networks. Such applications require receiver playback buffers to smooth network delay variations and to reconstruct the periodic nature of the transmitted packets. Packets arriving after their scheduled deadline are considered late and are not played out. Existing algorithms used in the Internet operate by adaptively adjusting the playout delay from talkpurt to talkpurt. There is an important tradeoff between loss rate and average playout delay. However, current algorithms fail to achieve a particular loss rate. Controllig this parameter is thus an important characteristic in any playout adaption algorithm. This paper presents a moving average algorithm for playout delay adaptation with tunable loss rate. Our trace-based simulations show that our algorithm performs very well in loss ranges of interest for interactive audio applications.

1 Introduction

Delay, jitter and packet loss in packet-switched wide-area networks are the main factors impacting audio quality in interactive multimedia applications. Today's Internet operates still in a best-effort basis, and

*University of Nice-Sophia Antipolis, IEEE Member. E-mail: Victor.Ramos@ieee.org

†Corresponding author

‡INRIA Sophia Antipolis, France. E-mail: Chadi.Barakat@sophia.inria.fr

§INRIA Sophia Antipolis, France. Also with C.E.S.I.M.O. Universidad de los Andes, Venezuela. E-mail: Eitan.Altman@sophia.inria.fr

thus the impact of jitter, delay and packet loss must be alleviated by employing end-to-end control mechanisms.

Audio applications such as NeVoT [1] and Rat generate packets spaced at regular time intervals. The traffic generated by an audio source is divided in periods of activity, called *talkspurts*, and periods of silence. Silence periods are periods where no audio packets are transmitted. To be able to playout the received packets as faithfully as possible, a receiving application must try to reproduce the periodic nature of the stream.

Audio packets encounter variable delay while crossing the Internet, which is mainly due to the variable queuing time in routers. This results in a stream of audio packets at the receiver that is no more periodic. In order to play the received stream, an application must buffer the packets and play them out after a certain deadline to get again a periodic stream at the application level. Packets arriving after their corresponding deadline are considered lost 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. 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 between 150 ms and 400 ms and loss rates up to 10% are considered adequate [2].

This paper focuses on the tradeoff between loss and delay in playout delay adaptation algorithms. Using measurements of audio packet delay, we present a Moving Average algorithm that adaptively adjusts the playout delay at the beginning of talkspurts. To prove the efficiency of our algorithm, we compare it with earlier work done by Ramjee et al [3]. Delay spike detection is done as in [3] in order to improve the performance of the moving average algorithm. We present two possible algorithms to adapt the playout delay: an offline algorithm and an online one. The offline moving average algorithm serves as a reference for our work. This offline implementation of the algorithm tells us that we have to predict a function of the delay, rather than the delay itself. We will detail on this transformation of the process that we need to predict. Then, we will show how an online hybrid algorithm can be implemented by combining the ideas proposed by Ramjee with the moving average algorithm we propose in this paper.

One characteristic that most of the playout delay adaptation algorithms lack is the ability to fix the

loss rate to some a priori value. By changing a measure of variability, the algorithms proposed by Ramjee et al. can achieve different loss rates. However, there is no explicit relationship between the measure of variability that we can adapt in these algorithms and the average loss rate. The average loss rate can change from one audio session to another, even if this parameter is kept unchanged. Here lies the main contribution of our work. We propose two moving average algorithms to adaptively adjust the playout delay from talkspurt to talkspurt, given a packet delay trace and a desired average loss rate p. Our algorithms ensure that the average loss rate we obtain during the session is close, if not equal, to the value we desire. At the same time, our algorithms realize this average loss rate with a smaller average playout delay than the one we need to obtain the same average loss rate with the algorithms proposed by Ramjee et al. For practical loss rates (between 1% and 20%), we show using real packet audio traces that our algorithms behave considerably better than existing algorithms in terms of the average playout delay, and the ability to control the packet loss rate to some a priori value.

The remainder of this paper is as follows. Section 2 provides some additional background and describes the packet delay traces we used as well as the notation we will use throughout the paper. In Section 3, we describe some related work. Section 4 describes the performance measures we will use to assess the performance of our algorithms and compare it with Ramjee's algorithms. Section 5 describes our moving average algorithm for a given loss rate. Section 6 describes, based on results from Section 5, that we have to predict a function of the delay rather than the delay itself, and compares the performance of our moving average algorithm with earlier work. Finally, Section 7 concludes this paper.

2 Some background

As we discussed in Section 1, receivers use a buffer to smooth the incoming stream and delay the playout of packets to compensate for variable network delay. The playout delay can be either constant through the whole session or can be adaptively adjusted from one talkspurt to the next. Yet, another recent approach [4] has shown that by using a technique called *packet scaling* it is possible to adaptively adjust the playout delay from packet to packet while keeping the resulting distorsion within tolerable levels.

In this paper we focus only on the adaptive per-talkspurt approach for playout delay adaptation. This results in regenerated streams at receivers having elongated or compressed silence periods compared with the original stream. These changes in the duration of silence periods are not noticeable by the human ear if they are kept within tolerable small levels.

Figure 1 shows the different stages between packet generation at the sender, packet reception at the receiver application, and packet playout. Packet i of talkspurt k is sent at time t_k^i, it arrives at the receiver at time a_k^i and it is hold in the smoothing receiver's buffer until time p_k^i. Packets are equally spaced at sender by intervals of time of length Δ seconds. In Figure 1, a lost packet due to a late arrival is represented by a dashed line. A packet is considered lost if it arrives after its scheduled deadline p_i^k. We also note in Figure 1 how as a result of the playout adaptation algorithm, silence periods can be artificially elongated or compressed.

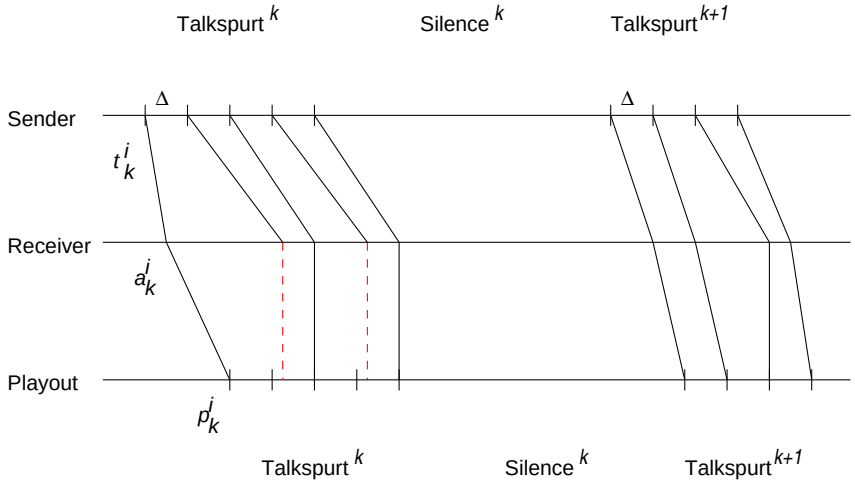

Figure 1: The timings between the transmission, reception and playout of packets.

Figure 1 shows how the stream of packets, when arriving at the receiver, losts its periodic nature. This is due to the variable delay introduced by the network when packets traverse intermediate routers. So, in order to reconstruct the original periodic form of the stream, packets are hold in the smoothing buffer at the receiver and their playout is delayed. An efficient playout adaptation algorithm must take into account the trade-off between between loss and delay, in order to keep both parameters as low as possible.

From here we will use the following notation.

- L : The total number of packets in 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 packet i of talkspurt k is generated at the sender.
- a_k^i : The time at which packet i 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. $d_k^i = a_k^i - d_k^i - \min_{1 \leq j \leq L} d_j$.
- p_k^i : The time at which packet i of talkspurt k is played out at the receiver.

We obtained publicly avalaible packet traces produced with the NeVoT audio tool [5]. NeVoT is an audio terminal program used to establish audio conversations in the Internet. The traces we use contain the sender and receiver timestamps of transmitted packets that are needed for the implementation of any playout adaptation algorithm. In the traces, one 160 byte audio packet is generated aproximately every 20 ms when there is speech activity. A description of each trace (reproduced from [5]) is given in Table 1. Table 2 gives additional statistics about these traces.

A typical period of packet delays is shown in Figure 2. A packet is represented by a diamond and talkspurt boundaries by a dashed rectangle. The x-axis represents the elapsed time since the beginning of the audio session at the receiver. Only the variable portion of the end-to-end delay is represented in Figure 2. To this end, the propagation component of the end-to-end delay is removed by substracting out from the trace the minimum measured end-to-end delay. Presumably, only in this way clock synchronization between sender and receiver can be avoided. Packets are generated at 20ms intervals, and hence a missing diamond in Figure 2 represents a lost packet in the network.

We also observe in Figure 2 the presence of delay spikes. This phenomenon in end-to-end delay was been extensively reported in the literature [6, 3]. Delay spikes are a serious problem in audio applications because 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 arriving almost

simultaneously, leading to the completion of the spike. In the left-hand side of Figure 2, a spike spans through 2 talkspurts and in the right-hand side we show a delay spike spanning over three talkspurts.

Delay spikes can be contained in a single talkspurt or can span through several talkspurts. Given that playout delay adaptation is done from talkspurt to talkspurt, a playout algorithm behaves better when delay spikes last more than a talkspurt than when a delay spike is properly contained within a talkspurt. Only in this way, a playout algorithm can react to the spike by setting the playout delay according to the experienced delay. If the spike vanishes before the end of a talkspurt, a playout algorithm would have no enough time to set the playout time accordingly.

In the next section, we briefly describe the algorithms proposed by Ramjee. Playout delay is adapted from talkspurt to talkspurt based in the past statistics of the delay process. The playout delay of the first packet of each talkspurt will dictate the deadlines for subsequent paquets in the same talkspurt. Generally speaking, this forms the basis of most of the existing playout adaptation algorithms.

Trace no.	Sender	Receiver	Start time	Length [s]	Multicast
1	UMass	GMD Fokus	08:41pm 6/27/95	1348	No
2	UMAss	GMD Fokus	09:58am 7/21/95	1323	Yes
3	UMAss	GMD Fokus	11:05am 7/21/95	1040	No
4	INRIA	UMass	09:20pm 8/26/93	580	No
5	UCI	INRIA	09:00pm 9/18/93	1091	No
6	UMass	Osaka Univ.	00:35am 9/24/93	649	No

Table 1: Description of the traces.

3 Related work

Extensive research work has been done in the subject of adaptive playout mechanisms [7, 8, 3, 4, 9, 5, 10]. In the current NeVoT audio tool, the authoring work of Ramjee et al is still implemented, and consequently, research on playout adaptation algorithms remains a very active research area. In [3] Ramjee

Trace	# of talkspurts	# of packets
1	818	56979
2	406	24490
3	536	37640
4	252	27814
5	540	52836
6	299	23293

Table 2: Some additional statistics of the traces considered.

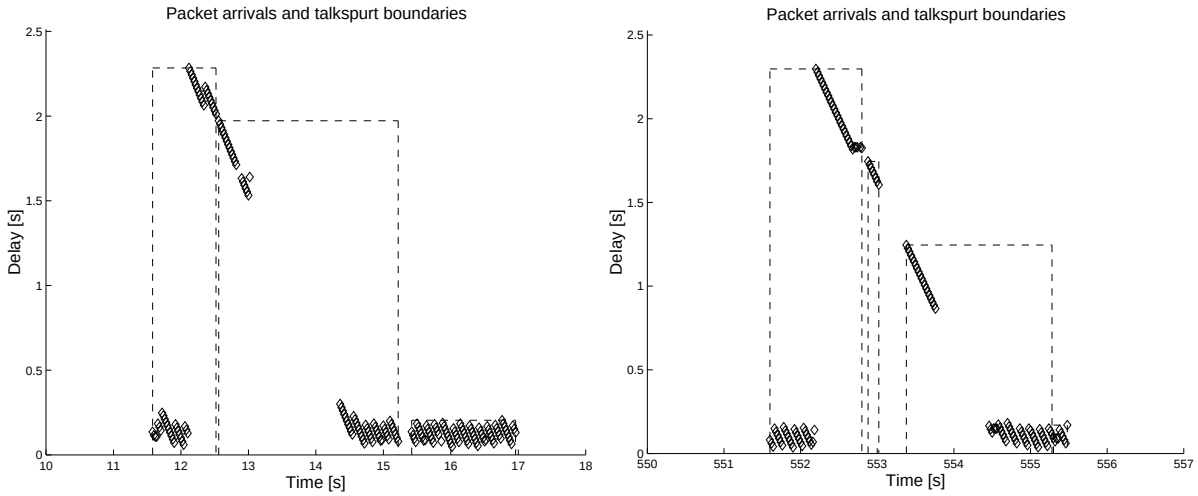

Figure 2: Delay spikes in end-to-end delay measurements.

et al. propose four algorithms for playout delay adaptation. All the four algorithms proposed in [3] use a delay estimate of the i-th packet as well as a measure of variation in delay similar to the calculation of round-trip-time estimates for the TCP retransmit timer. These statistics are used to compute the playout delay for a talkspurt.

The algorithms that perform better in [3] are algorithms 1 and 4. We rename here these algorithms as algorithm A and B respectively, and refer to them as such throughout the rest of this paper. Algorithm A is slow in detecting delay spikes but maintains a good overall average delay over an audio session. Algorithm B operates in two modes: normal mode and spike (or impulse) mode. When the difference between two consecutive delay values exceeds a given threshold, algorithm B triggers the spike mode.

When the end of spike is detected, algorithm B returns to normal mode.

To compute the playout delay of packet i of talkspurt k, algorithms both algorithms proceed as follows.

$$\begin{aligned} p_k^1 &= t_k^1 + \hat{d}_k^1 + \beta \hat{v}_k^1, \\ p_k^i &= p_k^i + (t_k^i - t_k^1), \end{aligned} \tag{1}$$

where $\hat{d}_k^1$ is an estimate of the mean end-to-end delay and $\hat{v}_k^1$ is an estimate of the variation in end-to-end delay. Both values are computed for each packet. Algorithm A computes $\hat{d}_k^i$ and $\hat{v}_k^i$ as follows :

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

To estimate the mean end-to-end delay $\hat{d}_k^i$ and the variability $\hat{v}_k^i$, algorithm B proceeds in a different way. It detects the beginning of a spike and it tries to follow it in order to react faster to sudden changes in the end-to-end delay. Algorithm B is described in [3] but it is repeated in Figure 3 for completness.

To compare our moving average algorithms with algorithms A and B, we have chosen to use the performance measures as defined in [5]. Next section defines such performance measures.

4 Performance measures

Let p_k^i be the playout delay of the i-th packet of talkspurt k. Let N be the total number of talkspurts in a trace, L the total number of packets in a trace, and let N_k the number of packets of talkspurt k. As in [5], we define r_k^i to be a variable indicating if packet i of talkspurt k has been played or not. So, r_k^i is defined as :

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

$d_i = a_i - t_i$;

```
if  (mode == NORMAL) {
```

```
    if (|d_i − d_{i−1}| > 2|v̂_i| + 800) {
```

```
       var = 0; /* Detected beginning of spike */
```

```
       mode = IMPULSE;
```

```
    }
```

```
}
```

```
else
```

```
    var = var/2 + |(2d_i − d_{i−1} − di − 2)/8|;
```

```
    if (var ≤ 63 ) {
```

```
       mode = NORMAL; /* End of spike */
```

$d_{i-2} = d_{i-1}$;

$d_{i-1} = d_i$;

```
       return;
```

```
    }
```

```
}
```

```
if
```

$d_i = 0.125 d_i + 0.875 \hat{d}_{i-1}$;

```
else
```

$\hat{d}_i = \hat{d}_{i-1} + d_i - d_{i-1}$;

$\hat{v}_i = 0.125 |d_i - \hat{d}_i| + 0.875 \hat{v}_{i-1}$;

$d_{i-2} = d_{i-1}$;

$d_{i-1} = d_i$;

```
return;
```

Figure 3: Algorithm B

In this way, the total number of packets , T, played out in a session is :

$$T = \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i. \tag{2}$$

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

$$D_{avg} = \frac{1}{T} \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i [p_k^i - t_k^i]. \tag{3}$$

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

$$l = \frac{L - T}{L} \times 100. \tag{4}$$

5 Moving Average prediction

The algorithms A and B proposed by Ramjee are good in maintaining a low overall average playout delay and reacting to delay spikes. However, they fail when a specific loss rate is desired given a packet delay trace. It would be desirable to have an algorithm to set the playout delay in a way that it allows to get a loss rate p given a trace of N talkspurts and L packets. By varying the parameter β in algorithms A and B, it is possible to obtain different loss rates, but one is unable to have any control on the overall loss rate p. We describe in this section a moving average predictor for playout delay that accepts a loss rate p and a packet delay trace as input, giving at its output a set of predicted playout delay values.

5.1 The Model

Let D_k be the optimal playout delay at the beginning of talkspurt k, and let p be the loss ratio during a talkspurt. We mean by *optimal playout delay* the delay level per-talkspurt allowing to obtain a given loss rate p. Let also N_k be the number of audio packets received during the k-th talkspurt and d_k^j the variable end-to-end delay of the j-th packet of talkspurt k. By sorting the set of playout delays $\{d_k^j\}$ in ascending order we set the *optimal playout delay* of talkspurt k to the following value :

$$D_k = d_k^i, \quad i \leq N_k, \tag{5}$$

with $i = \text{round}((1-p)N_k)$. So, if $d_k^i \leq D_k$ packet i is played out, otherwise the packet is dropped due to a late arrival.

We have a set of optimal past values $\{D_k, D_{k-1}, D_{k-2}, \ldots\}$ and we want to predict the value of D_{k+1}. The predicted value of D_{k+1} is denoted by $\hat{D}_{k+1}$, and it is taken as an *average* of the last M values of the process $\{D_k\}$. So

$$\hat{D}_{k+1} = \sum_{l=1}^{M} a_l D_{k-l+1}. \tag{6}$$

The coefficients a_l in (6) must be chosen in such a way that they minimize the mean square error between $\hat{D}_k$ and D_k, i.e. $\mathbb{E}[(\hat{D}_k - D_k)^2]$. The desired coefficients are the solution of the set of the so-called normal equations:

$$\sum_{m=0}^{M-1} a_m r_D(m-l) = r_D(l+1), \qquad l = 0, 1, \ldots, M-1. \tag{7}$$

In (7), r_D is the lag-l autocorrelation function of the process D_k, which is $\mathbb{E}[D_k D_{k+l}]$. The autocorrelation function can be estimated using the past values of the process. Suppose we have K values in the past, so we can write

$$r_D(r) \simeq \frac{1}{K-|r|} \sum_{k=1}^{K-|r|} D_k D_{k+|r|}, \quad r = 0, \pm 1, \pm 2, \ldots, \pm(K-1) \tag{8}$$

The set of normal equations (7) can be solved using the Levinson-Durbin recursion algorithm. M is called *prediction order* and its value can be determined in different ways. One way is to start with $M = 1$ and increment its value until an increase in the mean square error between $\hat{D}_k$ and D_k is noted. So, we take as the prediction order the lowest value of M preceding an increase in the mean square error. We choose to use this method to compute the prediction order of our moving average scheme.

5.2 Performance comparison

We will now compare the performance of our proposed moving average playout adaptation predictor with algorithms A and B. As pointed out above, with algorithms A and B we are unable to get a particular loss rate p. So, to get different loss rates we vary β from 1 to 20 in (1) and compare the performance of algorithms A and B with our moving average algorithm for loss rates between 1% and 20%. These loss rates are between the range of interest in real interactive audio sessions.

The computation of the playout delay for the k talkspurt is set as follows :

$$\begin{aligned} p_k^1 &= t_k^1 + \hat{D}_k \\ p_k^i &= p_k^i + (t_k^i - t_k^1). \end{aligned} \tag{9}$$

The moving average algorithm reacts to delay spikes that span more than one talkspurt as well as to delay spikes starting at the beginning of a talkspurt. In the moving average approach, delay spikes are handled as special cases. We use the same spike detection algorithm used in algorithm 4. Let d_k^i be the delay of packet i in the k-th talkspurt. If when starting talkspurt k the algorithm is in spike mode then the playout delay of talkspurt k is set to d_k^i, otherwise it is set to the value indicated by 9.

An important disadvantage of our Moving Average algorithm is that its execution depends on the knowledge of the process D_k which is unknown before the start of the session. So, the execution of this moving average approach can only be made offline. We see from Figure 4 how the moving average approach performs slightly better than algorithms A and B. However, the loss rate range from which it was intended our moving average approach is quite far from its expected bounds. The reason of this deviation is described in the next section. The results we obtained in this section will serve as a basis of a more accurate estimation approach. In the next section we also describe how a combination of Algorithm B and the moving average algorithm can give a very good performance and how it can be implemented online.

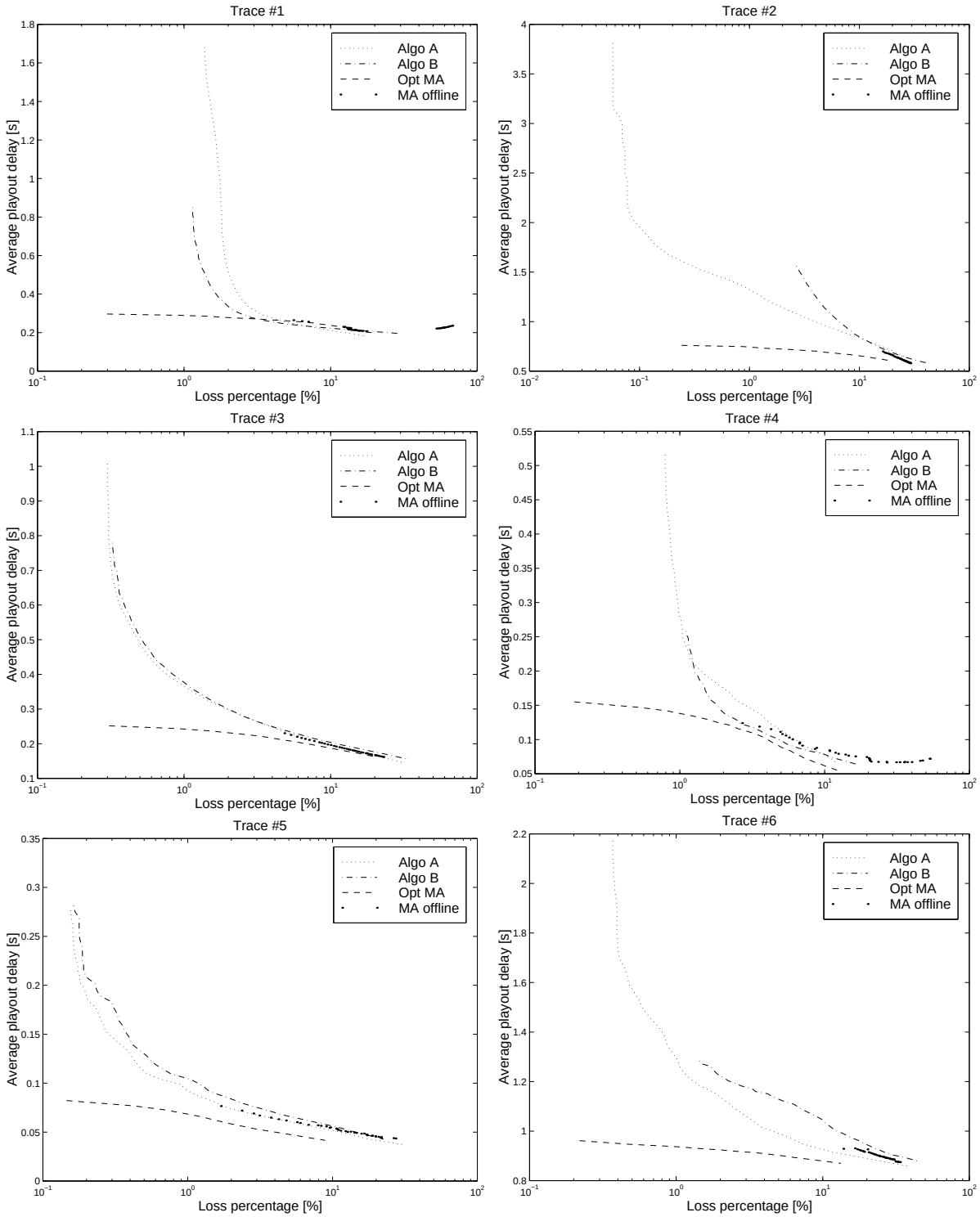

Figure 4: Performance comparison of algorithms A and B with MA.

6 Bias and Transformation

The above scheme improves the quality of the audio conversation by minimizing the loss ratio and the average playout delay. If we compare our scheme to existing ones, we notice that we can now obtain

the same loss ratio by using a smaller playout delay on average. This works very fine for relatively large loss ratios. However, the predictor fails to realize its objective when we set p, the desired loss ratio, to small values. First, when p is small, our scheme does not lead to the loss ratio we desire. Second, it does not lead to the considerable improvement of the average playout delay we are expecting from such an adaptive scheme using prediction techniques. We explain in this section this problem of our scheme, and introduce the necessary transformation that solves the problem and considerably improves the audio quality.

Our scheme is designed with the main objective to control the loss ratio at a certain value p, while minimizing the average playout delay. Here is the strength of our scheme, compared to the other schemes in the literature where we do not have a direct control on the loss ratio. Our control of the loss ratio has been done till now by controlling the optimal playout delay per talkspurt D_k. But, the relationship between the playout delay and the loss ratio is not linear. This is the reason why the loss ratio we obtain deviates from the desired one. The more the non-linearity of the relationship between the playout delay and the loss ratio, the more the deviation of our scheme from its objective. Technically speaking, our predictor is unbiased with respect to D_k, however it is biased with respect to the loss ratio per talkspurt. We illustrate the problem with the following example, which also points us to the required transformation.

Our moving average predictor of D_k ensures that $\mathbb{E}[D_k] = \mathbb{E}\left[\hat{D}_k\right]$. Let d_k^i, $1 \leq i \leq N_k$, be delay of the the i-th packet of talkspurt k. The way we define D_k also ensures that $\mathbb{E}\left[1\{d_k^i > D_k\}\right] = p$, with $1\{A\}$ being the indicator function. But, the average loss ratio we experience during the audio conversation is not $\mathbb{E}\left[1\{d_k^i > D_k\}\right]$, but rather $\mathbb{E}\left[1\{d_k^i > \hat{D}_k\}\right]$. Define the following function $\bar{F}(x) = \mathbb{E}\left[1\{d_k^i > D_k\} \,\middle|\, D_k = x\right]$. If d_k^i is independent of D_k (this might be the case when talkspurts are long), the function $\bar{F}(x)$ will be the complementary Cumulative Distribution Function (CDF) of the delay that audio packets experience in the network, i.e., $\bar{F}(x) = \mathbb{P}\left\{d_k^i > x\right\}$. We will come to this issue later when we propose an approximation for $\bar{F}(x)$. We have $\mathbb{E}\left[\bar{F}(D_k)\right] = p$. The expected loss ratio we measure during the conversation is $\mathbb{E}\left[\bar{F}(\hat{D}_k)\right]$. This measured loss ratio is the same as the desired one if the function $\bar{F}(x)$ is linear, since in this case $\mathbb{E}\left[\bar{F}(\hat{D}_k)\right] = \bar{F}\left(\mathbb{E}\left[\hat{D}_k\right]\right) = \bar{F}\left(\mathbb{E}\left[D_k\right]\right) = \mathbb{E}\left[\bar{F}(D_k)\right] = p$.

In reality, the function $\bar{F}(x)$ (can be seen as the tail of packet end-to-end delay) is not linear, and this

non-linearity explains why the measured loss ratio deviates from p. Suppose for example that $\bar{F}(x)$ is convex, and that the prediction error given D_k is unbiased, i.e., $\mathbb{E}\left[(\hat{D}_k) \,\middle|\, D_k\right] = D_k$ (with probability one), then

$$\mathbb{E}\left[\bar{F}(\hat{D}_k)\right] = \mathbb{E}\left[\mathbb{E}\left[F(\hat{D}_k)|D_k\right]\right] \geq \mathbb{E}\left[F\left(\mathbb{E}\left[\hat{D}_k|D_k\right]\right)\right] = \mathbb{E}\left[\bar{F}(D_k)\right] = p.$$

We use Jensen inequality combined with the fact that $\bar{F}(x)$ is supposed to be convex. In this convex case, the measured loss rate is larger than what we are expecting. Similarly, we can prove that the measured loss rate is smaller than what we are expecting if $\bar{F}(x)$ is a concave function.

To get rid of this bias caused by the non-linearity of $\bar{F}(x)$, we predict $\bar{F}(D_k)$, instead of predicting D_k. $\bar{F}(x)$ is the transformation to be made to the process $\{D_k\}$. Clearly, this requires the knowledge of the function $\bar{F}(x)$. Suppose for instance that this function is known. We introduce a new process $\{X_k\}$ defined as $X_k = \bar{F}(D_k)$. We apply the same prediction method (Moving Average of order M) to X_k. Denote by $\hat{X}_k$ the prediction of X_k. For past talkspurts, we compute first the optimal playout delay D_k, then we compute the value of X_k using the function $\bar{F}(x)$. We do the prediction, then we use the inverse of the function $\bar{F}(x)$ to set the playout delay at the beginning of the next talkspurt, i.e., we set the playout delay to $\bar{F}^{-1}(\hat{X}_k)$. By doing that, the measured loss ratio will be equal to the desired value p. Indeed,

$$\mathbb{E}\left[1\{d_k^i > \bar{F}^{-1}(\hat{X}_k)\}\right] = \mathbb{E}\left[\bar{F}(\bar{F}^{-1}(\hat{X}_k))\right] = \mathbb{E}\left[\hat{X}_k\right] = \mathbb{E}\left[X_k\right] = \mathbb{E}\left[\bar{F}(D_k)\right] = p.$$

We use for that the fact that our predictor is now unbiased with respect to X_k, that is $\mathbb{E}\left[\bar{X}_k\right] = \mathbb{E}[X_k]$. Before the transformation, our predictor was unbiased with respect to D_k.

We still have to choose the function $\bar{F}(x)$. We can compute this function offline for a long trace, or online using the history of the audio conversation. We choose to model the function $\bar{F}(x)$ using the complementary CDF of packet end-to-end delay. This leads to good performance given the large size of talkspurts (around X packets). The different traces we have studied show that $\mathbb{P}\left\{d_k^i > x\right\}$ decays exponentially with x, with sometimes a change in the decay rate. Based on this result, we adopt the following transformation: $\hat{F}(x) = e^{-\alpha x}$. The analysis we conduct on the different traces indicates that our scheme is not very sensitive to the value of α. We give then α the default value of 10 in Figure 6.

Figure 5 plots the cumulative distribution function of "optimal" delays $1 - F(D_k)$ (for $p = 0.05$) for the trace # 3. All the other traces have similar cumulative distribution functions.

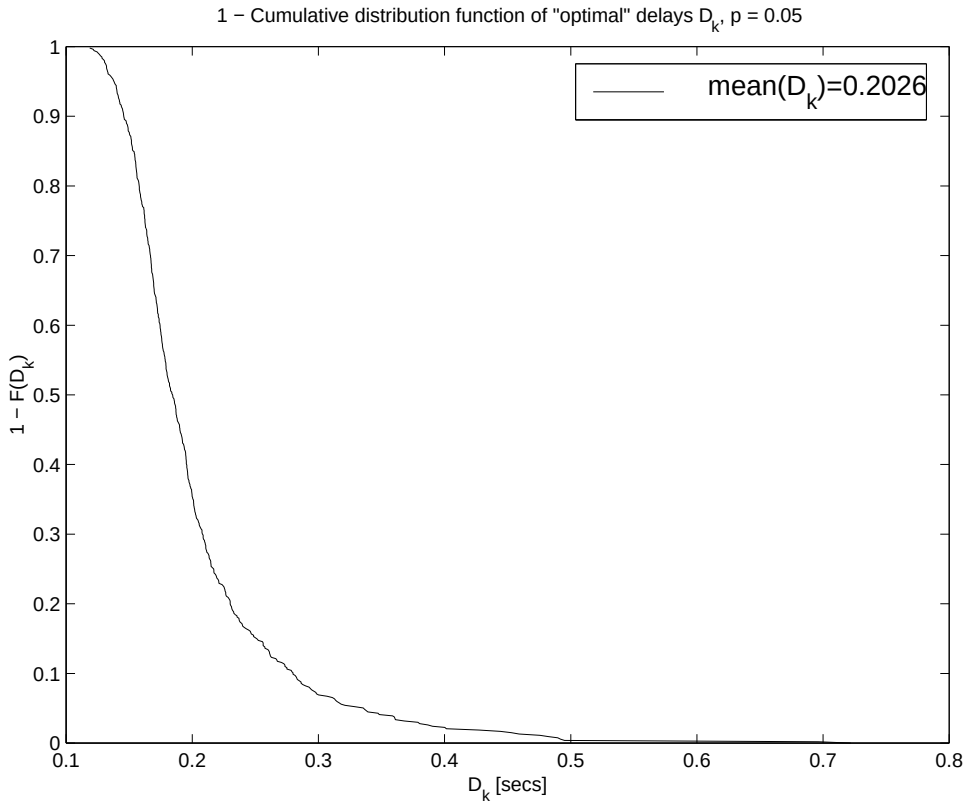

Figure 5: CDF of "optimal" delays D_k for trace 3.

6.1 Performance comparison

An online version of the moving average algorithm was implemented. To be able to implement our algorithm online, we executed a mixture of algorithms B and the moving average algorithm that we call *hybrid playout adaptation algorithm*. During the first part of the session (say Y talkspurts), the hybrid playout adaptation algorithm obtains enough statistics to compute predictions based on the history of the conversation. At the same time, the B algorithm is executed and playout delay values for the first Y talkspurts are set according to it. From talkspurt $Y + 1$ till the end of the session, the moving average algorithm just described is executed.

Figure 6 plots the performace of our hybrid algorithm compared to algorithms A and B. We see clearly

how, by using the transformation $\hat{F}(x) = e^{-\alpha x}$, with $\alpha = 10$ the overall average playout delay is kept as low as possible while having a loss rate p very near of its expected value. Here is the main advantage of our algorithm because we are able to set a given loss rate that is maintained for the last part of the conversation.

7 Conclusions

We described in this paper a moving average algorithm that adaptively adjusts the playout delay at the beginning of talkspurts. To evaluate the performance of our algorithm, we compared it with existing schemes implemented in the NeVoT audio tool. We presented two variants of our moving average algorithm : an offine algorithm and an online one. The offline implementation idicated us that instead of directly predicting the delay we have to predict a function of it.

The strenght of our scheme lies in the fact that we are able to tune the loss ratio p to a given desired value. When directly predicting the optimal playout delay, we observed that the desired loss ratio deviated from the desired one because the relation between the average playout delay and loss rate was not linear. We applied a transformation on the process of optimal delay D_k which we approached by a negative exponential function. A mixture of Ramjee's algorithm B and our moving average algorithm was implemented and it proved to be efficient in the loss ratios of interest.

7.1 Acknowledgment

The authors would like to thank Sue Moon for making publicily available the traces we used in our simulations.

References

[1] Henning Schulzrinne. Voice communication across the internet: a network voice terminal. Technical report, University of Massachusetts, Amherst., July 1992.

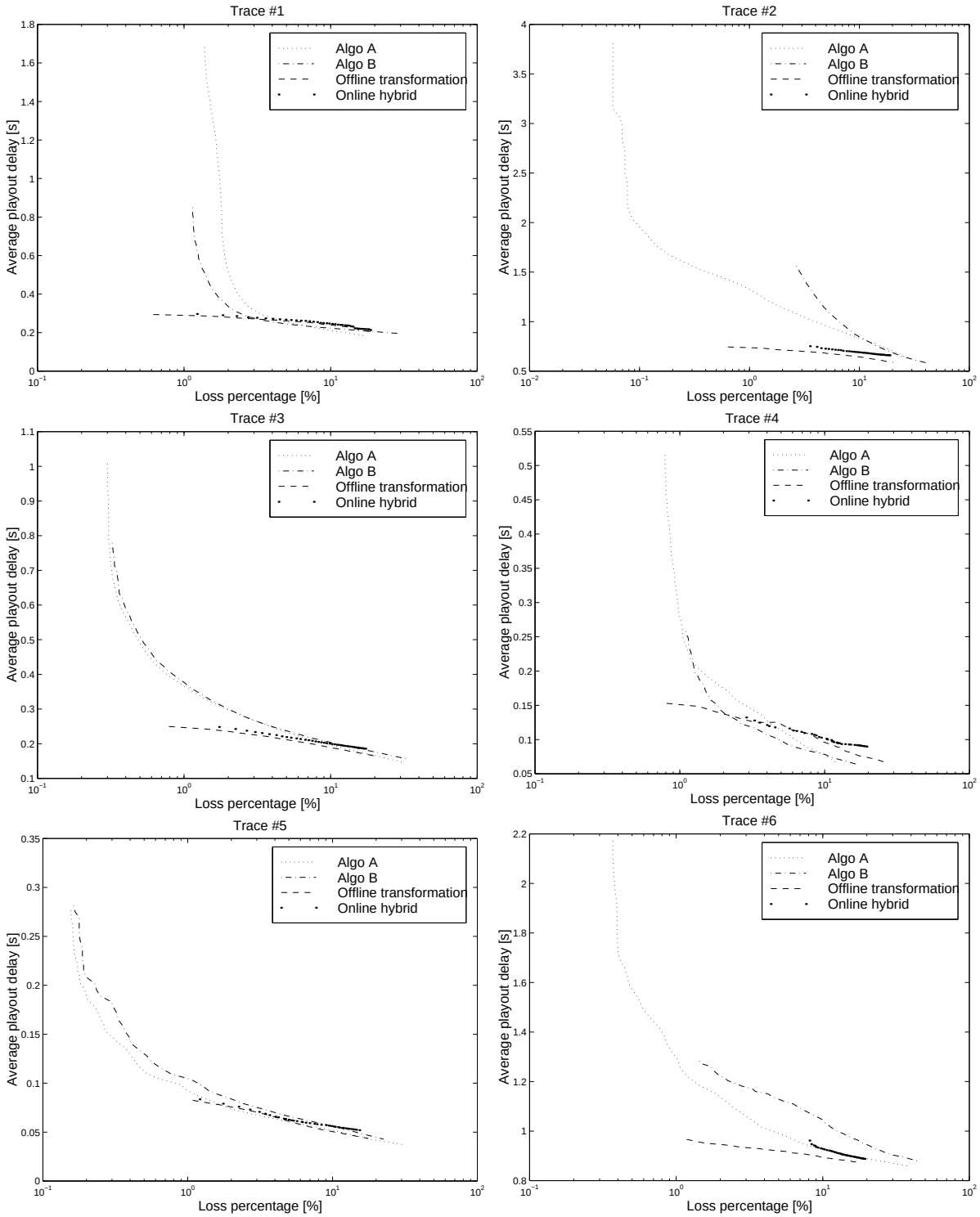

Figure 6: Performance comparison of algorithms 1 and 4 with our hybrid algorithm.

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

[3] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the IEEE Infocom*, pages 680–688, 1994.

[4] Yi J. Liang, Nikolaus Farber, and Bernd Girod. Adaptive playout scheduling and loss concealment for voice communications over IP networks. *IEEE Transactions on Multimedia*, April 2001.

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

[6] Jean Bolot. End-to-end packet delay and loss behavior in the internet. In *Proceedings of the ACM SIGCOMM*, pages 289–298, 1993.

[7] Nikolaus Farber, Yi Liang, Bernd Girod, and Balaji Prabhakar. Adaptive playout and TCP window control for voice over IP in best-effort networks. Technical report, Stanford University, Information Systems Laboratory, Department of Electrical Engineering, March 2001.

[8] Aman Kansar and Abhay Karandikar. Jitter-free audio playout over best effort packet networks. In *ATM Forum International Symposium*, New Delhi, India, 2001.

[9] Fang Liu, Jong Won Kim, and C.C. Jay Kuo. Adaptive delay concealment for internet voice applications with packet-based time-scale modification. In *Proceedings of the International Conference on Acoustics Speech and Signal Processing ICASSP*, Salt Lake City, May 2001.

[10] Jesús Pinto and Kenneth J. Christensen. 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, October 1999.