to appear in IEEE GLOBECOM 2002

Adaptive Playout Buffer Algorithm for Enhancing Perceived Quality of Streaming Applications

Kouhei Fujimoto †, Shingo Ata ‡, Masayuki Murata †,

†Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 560–8531, Japan
E-mail: {k-fujimo, murata}@nal.ics.es.osaka-u.ac.jp

‡Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan
E-mail: ata@info.eng.osaka-cu.ac.jp

This paper is intended to submit to Internet Performance Symposium 2002

Abstract— **The end-to-end packet delay is an important performance parameter on the Internet, because it heavily affects the quality of real-time applications. Currently, however, because the packet transmission quality (e.g., transmission delay, jitter, packet loss) may vary dynamically, it is not easy to handle real-time traffic. For UDP-based real-time applications, a smoothing buffer (playout buffer) is typically used at the client to compensate for variable delays. The issue of playout control has been studied previously, and several algorithms for controlling the playout buffer have been proposed. These studies considered the network parameters (e.g., packet loss ratio and playout delay), but not the quality perceived by end users.**

In this paper, we first clarify the relations between mean opinion score (MOS) of played audio and the network parameters (e.g., packet loss, packet transmission delay, and transmission rate). Then, utilizing the MOS function, we propose a new playout buffer algorithm that considers the user's perceived quality for real-time applications. Our simulation and implementation tests show that the algorithm can enhance the perceived quality, more effectively than existing algorithms.

I. Introduction

Due to the fast growth of the Internet, an increasing number of network applications are being used. These include real-time applications, such as IP telephony, voice conferencing, Internet radio, and video on demand (VoD), which have become widely used.

On the current Internet, however, because the packet transmission quality (e.g., transmission delay, jitter, packet loss) may vary dynamically, it is not easy to handle real-time traffic. For UDP-based real-time applications, a smoothing buffer is typically used at the client to compensate for variable delays. The received packets are first queued into the smoothing buffer. After several packets are queued, the actual decoding starts. Then, the influence of the delay variations within the network can be minimized. (We refer to this delay as the playout delay.) Choosing the playout delay is important because it directly affects the communication quality of the application. If the playout delay is set too short, the client application treats packets as lost even if they eventually arrive. On the other hand, a large playout delay may be unacceptably long so that the client users cannot tolerate it. Thus, it is difficult to determine the proper playout delay. The packet transmission delay between the server and client can be varied according to the network conditions on the Internet, so the appropriate playout delay heavily depends on the variations in the packet transmission delays. The issue of playout control has been studied previously [1], [2], [3], [4], and several algorithms for controlling the playout buffer (we refer to these as playout buffer algorithms (PBAs)) have been proposed. Most of these PBAs, however, are based on a calculation method using the time-out threshold in TCP [5]. For example, Moon et al. [3] trace the packet delays and suggested a playout delay based on the distribution of traced delays. However, they only focused on adjustments of the playout delay, and did not consider controlling the packet loss ratio (PLR).

In our prior work [6], we analyzed the characteristics of the packet transmission delays. We measured both the one-way transmission delay and the round-trip delay with synchronizing the measurement hosts by using Paxson's method [7] or global positioning system (GPS). From measurement results, we determined a suitable distribution function through a statistical analytic approach. We then introduced the use of the distribution function to estimate the playout delay for real-time applications. We proposed a new playout buffer algorithm, which manages the packet loss ratio according to the users' choices, while minimizing the playout delay.

However, neither the PLR nor the playout delay is a user-friendly metric for the perceived quality in streaming applications. There are many factors affecting the perceived quality of audio playback in streaming applications. Actually, in addition to the PLR, other network parameters such as the types of codecs, and the access lines also affect the perceived quality. One important issue is how to map these network metrics to the users' perceived quality with real-time traffic. Accordingly, we propose a new PBA to maximize the MOS index directly for given network parameters. Our approach utilizes the data set shown in [8], which clarified the relations between the MOS for played audio and the network parameters (e.g., packet loss, packet transmission delay, and transmission rate).

This paper is organized as follows. We first give a brief summary of existing PBAs and our prior work in Section II. In Section III, we examine the relations between the MOS and the

network parameters. Then, we propose a new PBA to maximize the MOS. In Section IV, we evaluate the proposed and existing algorithms through simulation and implementation. Finally, we summarize our work and discuss future research topics in Section V.

II. Introduction to Adaptive Playout Buffer Algorithms based on Network Parameters

In this section, we review some existing playout buffer algorithms for comparison with our proposed algorithm. Then we describe our prior work, in which we proposed an adaptive playout buffer algorithm based on analyzing packet delays.

A. Existing Playout Buffer Algorithms

For comparison purpose, we examine four algorithms which were proposed in [1], [3], and give brief overviews of each.

Exponential-Average (Exp-Avg): In this algorithm, the playout delay $\widehat{p}_i$ of the ith arriving packet is determined from approximated values for the mean $\widehat{d}_i$ and variance $\widehat{v}_i$ of the one-way delays, as given by

$$\widehat{p}_i = \widehat{d}_i + 4\widehat{v}_i, \quad (1)$$

$$\widehat{d}_i = \alpha\widehat{d}_{i-1} + (1-\alpha)n_i, \quad (2)$$

$$\widehat{v}_i = \alpha\widehat{v}_{i-1} + (1-\alpha)|\widehat{d}_i - n_i|, \quad (3)$$

where n_i denotes the one-way delay of the ith packet. The value of α is defined as 0.998002 according to [1]. Thus, the playout time $\widehat{t}_i$ is determined from the playout delay $\widehat{p}_i$ and the time s_i when the host sends the packet according to the equation; $\widehat{t}_i = \widehat{p}_i + s_i$. Here, the playout time means the time when the client actually starts playing the audio data recorded in the packet.

Thus, **Exp-Avg** estimates the playout time from means and variances, and does not consider the distribution of the delays.

Fast Exp-Avg (F-Exp-Avg): This algorithm is a modified version of **Exp-Avg**. **F-Exp-Avg** computes the weighted mean of $\widehat{d}_i'$s as

$$\widehat{d}_i' = \begin{cases} \beta\widehat{d}_{i-1}' + (1-\beta)n_i & \text{if } n_i > \widehat{d}_{i-1}', \\ \alpha\widehat{d}_{i-1}' + (1-\alpha)n_i & \text{otherwise,} \end{cases} \quad (4)$$

where α and β are constant values, satisfying $0 < \beta < \alpha < 1$. We set $\alpha = 0.998002$ and $\beta = 0.750000$ according to [1].

Spike Detection (SPD): This algorithm focuses on *spikes*, which represents sudden and large increases in delay over a sequence of a number of packets. Examples of spikes are shown at 3,850 in Figure 4(a). **SPD** usually obtains the playout delay from Eq. (2), which is the same as **Exp-Avg**. During a spike, however, **SPD** uses the following equation; $\widehat{d}_i = \widehat{d}_{i-1} + n_i - n_{i-1}$, which accounts for the sudden increase in delay. For **SPD**, we use $\alpha = 0.875$ according to [1].

Window: This algorithm, proposed in [3], is designed to detect spikes like **SPD**. During a spike, the first packet in the spike is used as the playout delay. After the spike, the playout delay is chosen by finding the delay corresponding to the qth quantile of the distribution of the last N packets received by the client. In our evaluation, a value of 0.99 is used for q, and 10,000 is used for N, as described in [3].

B. Prior Work

To provide high-quality communication for streaming applications, it is desirable for the packet loss ratio and playout delay to be kept small. However, there is a critical trade-off between the packet loss ratio and the length of the playout delay.

Hence, in our prior work [6], [9], we measured packet transmission delays and analyzed their characteristics by taking the network parameters into account. We then proposed a method of modeling the tail distributions of the delays, which is available for applications. From the results of this statistical analysis, we found that the Pareto distribution is most appropriate as a model of the one-way delay distribution under any network conditions. The Pareto distribution is widely known to be able to represent a self-similarity [10], whose cumulative distribution function (CDF) is given by

$$F(x) = 1 - \left(\frac{k}{x}\right)^{\alpha}, \quad x \geq k. \quad (5)$$

where α and k are the parameters of a Pareto cumulative distribution function. Next, we proposed a new playout buffer algorithm based on our statistical analysis. The proposed algorithm determines the playout delay so as to provide the packet loss ratio specified by the users.

Here we discuss the design of our proposed playout buffer algorithm more specifically. The algorithm records the history for the one-way delays of packets. Upon each packet arrival, the parameters (k, α) of the Pareto cumulative distribution function $F(x)$ are updated to estimate the playout delay p_i from the equation $F(p_i) = X$, where X is a target value. The target value is the reproduction ratio of packets specified by the user. From the Pareto CDF, our proposed algorithm determines the playout delay as

$$\widehat{p}_i = \frac{k}{\sqrt[\alpha]{1 - \frac{X}{100}}}. \quad (6)$$

We consider 95, 99, and 99.9% as target values X based on our numerical results. We refer to this proposed playout buffer

algorithm as the loss-control playout buffer algorithm (**Loss-Control**). Numerical examples have shown that **Loss-Control** can control the playout buffer while providing the target packet loss probability. Section 4 includes evaluation results for this control method, including results for our new algorithm described in the next section.

III. Proposed Playout Buffer Algorithm to Maximize Perceived Quality

In our previous work [6], we demonstrated that **Loss-Control** can control the packet loss ratio according to the users' choice. However, there is a high possibility that the delay and other network parameters (types of codecs, access lines) would affect the perceived quality, in addition to the PLR. The end users can choose their preferred playout quality, but there are still many other parameters left for them to configure. It is hence necessary to introduce a simpler index that directly relates to the perceived quality of multimedia communications. Today, many metrics expressing the playout quality have been proposed and evaluated. The subjective metrics that we adopt in this paper are more user-friendly because they are based on scores determined by users according to their experiences listening to or watching various media.

Our objective in this section is to maximize the subjective index of the perceived quality for given network parameters, which are automatically measured. We first model the relations between the MOS and the network parameters shown in [8] in terms of mathematical formulas. After modeling, we obtain the MOS-relative form, which gives the appropriate packet loss ratio and playout delay according to the MOS value. We then modify our **Loss-Control** PBA by applying this MOS-relative function. Numerical comparisons are shown in the next section.

A. Effects of Packet Loss Ratio and Delay on MOS

To clarify the relations between the MOS value and the network parameters, we take data from [8], which shows the effects of the network parameters on the MOS. We show one such piece of data in Figure 1. Each plotted curve shows a relation between MOS and end-to-end one-way delay for a given loss ratio. From the model of the one-way delay distribution described in our previous work [6], we can obtain feasible combinations of the playout delay and the packet loss ratio to maximize the MOS index. We describe our modeling method in the next subsection.

B. Modeling Methods for MOS Functions

The first step in our modeling method is to formulate approximate relations among the MOS, packet loss ratio, and playout delay. That is, we plot the MOS curves shown in Figure 1 by using mathematical notations based on our modeling method. Of course, the resulting formulas depend on the data shown in Figure 1, but our modeling approach is also applicable other results.

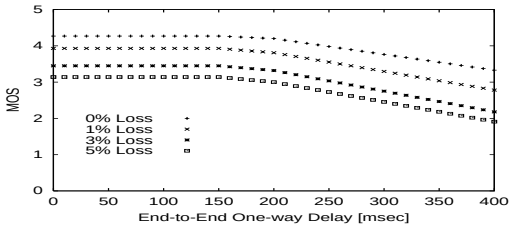


Fig. 1. Effects of PLR and Delay (Encoder: G.711)

From Figure 1, we can obtain the following assumptions.

- The four curves shown in the figure are parallel. This means that the packet loss ratio and one-way delay, and hence the playout delay, affect MOS independently. Therefore, we can separately consider the effects of packet loss ratio and one-way delay on modeling MOS.
- The degree of degradation in MOS values is proportional to the packet loss ratio, and does not depend on the playout delay.

Given these assumptions, we can obtain the MOS function $M(p,d)$ for a given packet loss ratio p and playout delay d from $M(p)$ and $M(d)$ separately. We now determine the MOS function as follows.

We first model the MOS function $M(d)$ for a given playout delay d with a three-dimensional polynomial approximation, where the packet loss ratio p is assumed to be zero (shown by the crosses in Figure 1). The coefficients of the polynomial are obtained by curve fitting. We then obtain the MOS function $M(d)$ as

$$\begin{aligned} M(d) &\approx 4.10 + 2.64 \times 10^{-3} d - 1.86 \times 10^{-5} d^2 \\ &\quad + 1.22 \times 10^{-8} d^3. \end{aligned} \tag{7}$$

We then obtain the MOS curve by sliding inversely in the horizontal direction. Based on our second assumption above, the degree of degradation of the MOS values is proportional to the packet loss ratio. We calculate the parameters of the function by the least linear square method. The MOS function $M(p)$ for a given packet loss ratio p is thus expressed as

$$M(p) \approx 4.10 - 0.195p, \tag{8}$$

where the playout delay is set to $d = 0$.

Because we assume that the packet loss ratio and the playout delay affect the MOS value independently, we can obtain

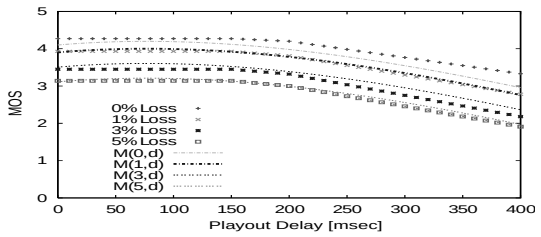


Fig. 2. Results of Modeling MOS values and Network Parameters (Encoder: G.711)

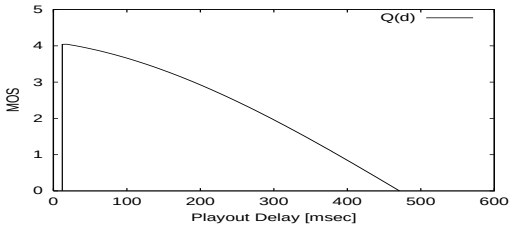


Fig. 3. MOS Function $Q(d)$ (Encoder: G.711)

$M(p, d)$ for given p and d by combining Eqs. (7) and (8), i.e.,

$$M(p,d) = 4.10 - 0.195p + 2.64 \times 10^{-3}d - 1.86 \times 10^{-5}d^2 + 1.22 \times 10^{-8}d^3. \quad (9)$$

In Figure 2 we add the solid curves representing Eq. (9) to the curves in Figure 1, and we can observe that our approximate modeling agrees with the original data.

In a real network, however, there is a correlation between the packet loss ratio p and the playout delay d. In streaming applications, the packet loss ratio p is a summation of (1) the packet loss ratio caused by packet drops within the network (referred to as p_n), and (2) the ratio of late-arriving packets exceeding the playout threshold (p_d). That is, $p = p_n + p_d$. From Eq. (5) in Section II, we determine the relation between p_d and d as follows:

$$p_d = 100\left(\frac{k}{d}\right)^{\alpha}. \quad (10)$$

By applying Eq. (10), Eq. (9) can then be rewritten as

$$M(p_n,d) = 4.10 - 0.195\left(p_n + 100\left(\frac{k}{d}\right)^{\alpha}\right) + 2.64 \times 10^{-3}d - 1.86 \times 10^{-5}d^2 + 1.22 \times 10^{-8}d^3. \quad (11)$$

As shown by Eq. (11), the two parameters d and p_n affect the MOS value. For streaming applications, however, only the playout delay d is controllable. We therefore redefine Eq. (11) as a function of d, denoted as $Q(d)$, i.e.,

$$Q(d) = 4.10 - 0.195p_n - 19.5\left(\frac{k}{d}\right)^{\alpha} + 2.64 \times 10^{-3}d - 1.86 \times 10^{-5}d^2 + 1.22 \times 10^{-8}d^3. \quad (12)$$

We now examine $Q(d)$. If $d = 0$, all packets are treated as lost, and no packet is played. Thus, we set $Q(0) = 0$. As the playout delay is increased, $Q(d)$ takes larger values. However, when the playout delay is too large, $Q(d)$ is again degraded due to the large delay for playback. Therefore, there is an optimum d that produces the maximum value of $Q(d)$. Figure 3 shows an example of variation in $Q(d)$ dependent on the playout delay d, where α and k in Eq. (12) are set to 9.10 and 15.53, respectively, based on measured data. The optimal d is calculated by the *false position method* [11] utilizing a differential equation of $Q(d)$. Because $Q(d)$ is a convex function, we can use the false position method to determine the optimal d from the x-intercept of the differential equation of $Q(d)$.

C. Modified Playout Buffer Algorithm for Enhancing MOS Index

We modified our **Loss-Control** PBA to achieve MOS-based control. In the **Loss-Control** algorithm, the playout delay is determined from the target packet loss ratio. On the other hand, our new algorithm controls the playout delay by maximizing the MOS value $Q(d)$. More specifically, our new PBA consists of the following steps;

1. Measure the transmission delays of arriving packets
2. Calculate the parameters of the Pareto distribution (α, k) by the MLE method (see our prior work [6])
3. Use the values of (α, k) in the MOS function $Q(d)$
4. Obtain the optimal value of d to maximize $Q(d)$, by applying the false position method to the differential equation of $Q(d)$
5. Set the playout delay to d
6. Return to Step 1

We refer to this new playout buffer algorithm as the enhanced MOS-based playout buffer algorithm (**E-MOS**).

IV. Evaluation of Playout Buffer Algorithm

In this section, we evaluate the playout buffer algorithms by trace-driven simulation, and we investigate the effectiveness of our proposed algorithm.

A. Simulation Method

We prepared a set of one-way delays of packets for our trace-driven simulation. For this purpose we measured the one-way delays with various network parameters. In the simulation, the

TABLE I
COMPARISON OF PLR AND MEAN PLAYOUT DELAY AND MOS

Case	Algorithm	Target	PLR [%]	Mean of d_i [msec]	MOS
"dynamic"	Loss-Control	95%	5.7	227.92	2.22
		99%	0.94	387.12	2.41
		99.9%	0.12	770.44	0.59
	E-MOS	-	2.95	294.75	**2.49**
	Exp-Avg	-	4.54	247.91	2.38
	F-Exp-Avg	-	0.1	970.34	0.10
	SPD	-	5.44	198.74	2.33
	Window	99%	1.34	362.57	2.47
"moderate"	Loss-Control	95%	6.02	40.61	2.99
		99%	1.77	58.45	3.83
		99.9%	0.60	375.28	3.61
	E-MOS	-	0.10	77.71	**4.17**
	Exp-Avg	-	4.93	39.79	3.21
	F-Exp-Avg	-	0.04	102.26	4.13
	SPD	-	3.08	39.74	3.57
	Window	99%	2.33	48.60	3.72
"quiet"	Loss-Control	95%	3.94	9.40	2.94
		99%	0.72	9.87	3.60
		99.9%	0.22	10.53	3.70
	E-MOS	-	0.00	51.92	**3.77**
	Exp-Avg	-	0.18	10.49	3.71
	F-Exp-Avg	-	0.01	29.53	3.76
	SPD	-	0.77	10.19	3.59
	Window	99%	1.05	9.76	3.53

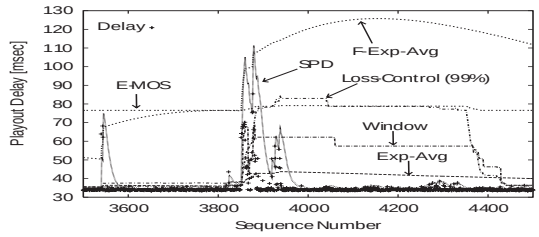
recorded one-way delays are used one-by-one, and the playout delay p_i of the ith packet is estimated according to each algorithm for all the measured delays. Then, we check whether the delay of the next packet is smaller than the estimated playout delay. If it is larger than the estimated playout delay, the packet is treated as lost. After tracing all the measured delays, the average playout delay and packet loss ratio are computed as the output.

B. Simulation Results

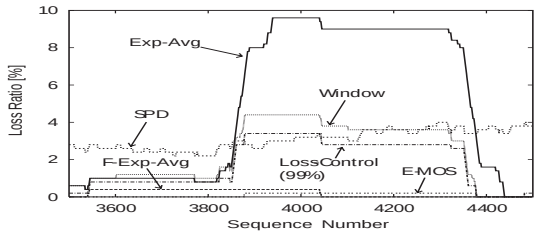
Table I compares the PLRs, mean values of the playout delays, and MOS evaluated by simulation for three different cases. The first case is "dynamic", in which the values of the one-way delays often change, and many spikes are observed. The packets were sent by the G.723.1 encoder at 2 PM and delivered to the receiver via a dial-up line. The second case is a "moderate", in which there are several spikes. We used the G.711 encoder over ADSL and the delays were measured at 1 PM. The last case is "quiet", in which no dynamic changes in the delays are observed. These delays were sent by the G.723.1 encoder at 2 PM and delivered to the receiver over a LAN. In **Loss-Control**, we use 95, 99, and 99.9% as the target values. The MOS values shown in the last column of Table I were evaluated by Eq. (9) from the PLR and the playout delay. The maximum MOS values among all the PBAs are shown in bold.

The results in Table I indicate that **E-MOS** can provide the highest perceived quality for users under any network conditions. Looking at the playout delays and PLRs of **E-MOS**, we found that it has a tendency to minimize the PLR when the one-way delays are small (the moderate and quiet cases in Table I). From Figure 1, we can observe that the effect of introducing the playout delay is quite limited when the delay is small (less than 200 msec). In this region, it is effective to prevent packet loss by lengthening the playout delay. However, as the one-way delay becomes larger, **E-MOS** tries to intentionally accommodate the increasing PLR to reduce the playout delay. This is a good solution to improve the users' perceived quality. Other PBAs have their own approaches. For example, **Window** gives a good result in the dynamic case but is worse than **F-Exp-Avg** in other cases. However, **F-Exp-Avg** performs quite poorly in the dynamic case. **Exp-Avg** and **Loss-Control (99.9%)** perform passably in all cases. However, these methods cannot attain the same improvement in the perceived quality as **E-MOS**. Furthermore, the **Loss-Control** method has a disadvantage in that it tries to shorten the playout delay and forces the abandonment of packets even when the playout delay is sufficiently short (less than 200 msec). Thus, **Loss-Control** is not suitable for low packet transmission delay environments.

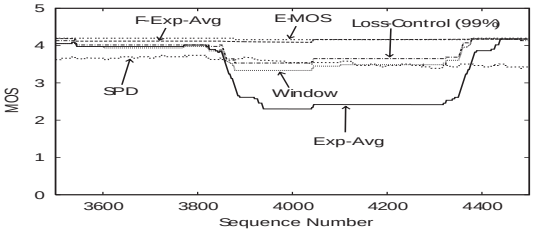
Figure 4 shows the time-dependent behavior of the playout delay, PLR, and MOS for each PBAs. Here, the target value of **Loss-Control** is set to 99%. The playout delays of **E-MOS** are



(a) Comparison of Playout Delay



(b) Comparison of PLR



(c) Comparison of MOS

Fig. 4. Performance Evaluation of Each PBA ("moderate" case)

larger than those of the other algorithms except for **F-Exp-Avg**, for which the one-way delays are small. From these results, we found that **E-MOS** tends to minimize the PLR when the one-way delay is less than 200 msec. On the other hand, when the one-way delay is over 200 msec, **E-MOS** tries to increase the PLR to reduce the playout delay and thus enhance the MOS. That is, **E-MOS** can achieve a good balance between the playout delay and PLR based on Eq. (12).

C. Evaluation through Implementation Experiments

We developed a streaming client on which our PBA was implemented, and we verified the applicability of our algorithm by running the application. More specifically, we implemented our PBA as an input plug-in for `Winamp` [12], which is currently one of the major front-end real-time applications.

Fig. 5. Operation Window of Client†

We set up the streaming server at Osaka University. The server sent audio packets generated by the G.711 or G.728 encoders (the sizes of the packet and transmission intervals are 160 bytes and 20 msec for G.711 and 40 bytes, 20 msec for G.728, respectively). The packets were transmitted via the Internet to the client we developed. On the client, the smoothing buffer was adjusted based on the playout delay calculated by our PBA (**E-MOS**). The arriving packets were stored in the buffer, and then the client started playback after the playout interval. Figure 5 shows the operation window of our client.

Our implementation experiments included (1) checking whether our PBA tries to maximize the MOS, and (2) verifying whether the computational overhead of calculating the playout delays is sufficiently small to operate our PBA in *real-time*.

The platform was the Microsoft Windows 98 operating system on an Intel Pentium III 750-MHz CPU. With this platform, the computation overhead was about 0.02 msec for each packet arrival; this is 0.1% of the packet transmission interval for G.711, which is sufficiently small overhead. We also confirmed that the audio playback was not interrupted by any factors other than packet losses.

V. Concluding Remarks

In this paper, we have considered the perceived quality of streaming applications and modified our previously proposed algorithm so as to maximize the perceived quality. Simulation and implementation experiments have shown that the modified algorithm provides the highest quality of any PBA.

For future research, it will be necessary to improve the accuracy of our model representing the delay distributions. To achieve this, it might be useful to test other heavy-tailed probability functions as potential models for the delay distributions. Moreover, although no serious problems occur at the client with **E-MOS**, a smaller CPU load would be more efficient for users. A more effective calculation method for **E-MOS** is thus necessary.

† ©Nullsoft Inc. 2002

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 INFOCOM '94*, pp. 680–688, April 1994.

[2] B. J. Dempsey and Y. Zhang, "Destination buffering for low-bandwidth audio transmissions using redundancy-based error control,," in *Proceedings of LCN, 21st Annual Conference on Local Computer Networks*, pp. 345–354, October 1996.

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

[4] S. Mohamed, F. Cervantes-Pérez, and H. Afifi, "Integrating networks measurements and speech quality subjective scores for control purpose," in *Proceedings of IEEE INFOCOM 2001*, April 2001.

[5] J. Postel, "Transmission control protocol specification," *RFC 793*, September 1981.

[6] K. Fujimoto, S. Ata, and M. Murata, "Statistical analysis of packet delays in the Internet and its application to playout control for streaming applications," *IEICE Transactions on Communications*, vol. E84-B, pp. 1504–1512, June 2001.

[7] V. Paxson, "On calibrating measurements of packet transit times," in *Proceedings of ACM SIGMETRICS '98*, pp. 11–21, June 1998.

[8] C. Savolaine, "QoS/VoIP overview," in *IEEE Communications Quality & Reliability (CQR 2001) International Workshop*, April 2001.

[9] K. Fujimoto, S. Ata, and M. Murata, "Playout control for streaming applications by statistical delay analysis," in *Proceedings of IEEE International Conference on Communications (ICC 2001)*, vol. 8, (Helsinki), pp. 2337–2342, June 2001.

[10] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web; traffic evidence and possible causes," in *Proceedings of ACM SIGMETRICS '96*, pp. 160–169, May 1996.

[11] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C; The Art of Scientific Computing*, ch. 9.2, pp. 263–266. Cambridge University Press, 1988.

[12] NULLSOFT, "WINAMP.COM | now featuring self-transforming mechanical elves." available at `http://www.winamp.com`.