Adaptive Playout Buffer Algorithm for Enhancing Perceived Quality of Streaming Applications

Kohei Fujimoto † **Shingo Ata** ‡ **Masayuki Murata** §

†Graduate School of Engineering Science, Osaka University

1-3 Machikaneyama, Toyonaka, Osaka 560–8531, Japan

Phone: +81–6–6850–6588, Fax: +81–6–6850–6589

E-mail: k-fujimo@nal.ics.es.osaka-u.ac.jp

‡Graduate School of Engineering, Osaka City University

3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan

Phone, Fax: +81–6–6605–2191

E-mail: ata@info.eng.osaka-cu.ac.jp

§Cybermedia Center, Osaka University

1–30, Machikaneyama, Toyonaka, Osaka 560-0043, Japan

Phone: +81-6-6879-8793, Fax: +81-6-6879-8794

E-mail: murata@cmc.osaka-u.ac.jp

Abstract

An end-to-end packet delay in the Internet is an important performance parameter, because it heavily affects the quality of real-time applications. In the current Internet, however, because the packet transmission qualities (e.g., transmission delays, jitters, packet losses) may vary dynamically, it is not easy to handle a real-time traffic. In UDP based real-time applications, a smoothing buffer (playout buffer) is typically used at a client host to compensate for variable delays. The issue of playout control has been studied by some previous works, and several algorithms controlling the playout buffer have been proposed. These studies have controlled the network parameters (e.g., packet loss ratio and playout delay), not considered the quality perceived by users.

In this paper, we first clarify the relationship between Mean Opinion Score (MOS) of played audio and network parameters (e.g., packet loss, packet transmission delay, transmission rate). Next, utilizing the MOS function, we propose a new playout buffer algorithm considering user's perceived quality of real-time applications. Our simulation and implementation tests show that it can enhance the perceived quality, compared with existing algorithms.

All correspondence should be directed to:

Dr. Shingo Ata

Department of Information and Communication Engineering

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

Phone, Fax: +81–6–6605–2191

1 Introduction

According to the fast growth of the Internet, there are an increasing number of network applications. The real-time application is one of such applications. The use of real-time application includes IP telephony, voice conference, internet radio, video on demand (VoD). Now, these applications become widely used.

In the current Internet, however, because the packet transmission qualityies (e.g., transmission delays, jitters, packet losses) may vary dynamically, it is not easy to handle a real-time traffic. In UDP based real-time applications, a smoothing buffer is typically used at a client host to compensate for variable delays. Received packets are first queued into the smoothing buffer. After several packets are queued, actual decoding is started. Then, the influences 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 to be too short, the client application treats packets to be lost even if those packets eventually arrive. On the contrary, the large playout delay may introduce an unacceptable delay that the client users cannot be tolerant. That is, a difficulty exists in determining the playout delay. The packet transmission delay between the server and client may be varied according to the network condition in the Internet, and hence, the adequate playout delay is heavily dependent on variations of packet transmission delays. The issue of playout control has been studied by some previous works [1, 2, 3, 4], and several algorithms controlling the playout buffer (which we will refer to as the playout buffer algorithm (PBA)) have been proposed. Most of those PBAs are however based on a calculation method of the time-out threshold in TCP [5]. For example, Moon et al. [3] trace the packet delays and suggest the playout delay from the distribution of traced delays. However, they only focus on adjustments of the playout delay, and do not consider to control the packet loss ratio.

In our prior work [6], we analyzed the characteristics of the packet transmission delays. We measured the one-way transmission delay as well as the round-trip delay, and provided the determination of the suitable distribution function through a statistical analytic approach. We next introduced the use of the distribution function to estimate the playout delay for real-time applications. we proposed a new playout buffer algorithm, which keeps the packet loss ratio according to users' willingness while minimizing the playout delay.

However, neither the packet loss ratio nor playout delay is not a user-friendly metric for the perceived quality in streaming applications. There are many factors affecting the perceived quality in playing audio or speech of the streaming applications. Actually, in addition to the packet loss ratio, other network parameters such as type of codecs, access lines would affect the perceived quality. One important issue is how to map the network metrics into the users' perceived quality of the real-time traffic. Accordingly, we propose a new PBA to maximize the MOS index directly for given network parameters. Our approach is to utilize the data set shown in [7], which clarified the relationship between Mean Opinion Score (MOS) of played audio and network parameters (e.g., packet loss, packet transmission delay, and transmission rate).

This paper is organized as follows. We first show a brief summary of existing PBAs and our prior work in Section 2. In Section 3, we examine the relationship between MOS of played audio and network parameters. Then, we propose a new PBA in order to maximize the MOS. In Section 4, we evaluate the proposed algorithms through the simulation and implementation. Finally, we summarize our work and describe our future research topics in Section 5.

2 Introduction of Adaptive Playout Buffer Algorithms based on Network Parameters

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

2.1 Existing Playout Buffer Algorithms

For comparison purpose, we examine four algorithms which have been proposed in [1, 3]. Before introducing our proposed playout buffer algorithm in prior work, we describe the brief overviews of each playout buffer algorithm.

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

$$\begin{aligned} \widehat{p}_i &= \widehat{d}_i + 4\widehat{v}_i, & (1) \\ \widehat{d}_i &= \alpha\widehat{d}_{i-1} + (1-\alpha)n_i, & (2) \\ \widehat{v}_i &= \alpha\widehat{v}_{i-1} + (1-\alpha)|\widehat{d}_i - n_i|, & (3) \end{aligned}$$

where n_i means the one-way delay of ith packet. The value of α is chosen to be 0.998002 according to [1]. Thus, playout time $\widehat{t}_i$ is determined from playout delay $\widehat{p}_i$ and time s_i when the sender host sent packet according to the equation; $\widehat{t}_i = \widehat{p}_i + s_i$. Here, the playout time means the time when the application client actually starts playing 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

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

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

Spike Detection (SPD): This algorithm focuses on *spike* which represents a sudden and large increase in delays over a sequence 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 same as **Exp-Avg**. During spike, however, **SPD** uses the following equation; $\widehat{d}_i = \widehat{d}_{i-1} + n_i - n_{i-1}$, to catch up the sudden increase of delays. In **SPD**, we use $\alpha = 0.875$ following [1].

Window: This algorithm proposed in [3] intends to detect a spike as **SPD**. During the spike, the first packet in the spike is used as a playout delay. After 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 receiver. In our evaluation, a value of 0.99 is used for q, and 10,000 for N, which are used in [3].

2.2 Prior Work

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

In our prior work [6, 8], we hence measured packet transmission delays and analyzed their characteristics by taking into account the network parameters. We then proposed a method modeling the tail distribution of the delays, which is available for applications. From the results of statistical analysis, we found that the Pareto distribution is most appropriate as the model of one-way delay distribution in any network conditions. The Pareto distribution is widely known to be able to represent a self-similarity [9], whose cumulative distribution function (CDF) is given by

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

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 that the packet loss ratio specified by the users is satisfied.

We show the design of our proposed playout buffer algorithm more specifically. Our playout buffer algorithm records the history of one-way delays of packets. On each packet arrival, 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 means the reproduction ratio of packets preferred by the user. From the Pareto CDF, our proposed algorithm determines the playout delay by

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

We consider 95, 99, and 99.9% as the target value X through our numerical results. We refer to our proposed playout buffer algorithm as the loss control playout buffer algorithm (**Loss-Control**). Numerical examples showed that **Loss-Control** can control the playout buffer with satisfying the target packet loss probability. See Section 4 for evaluation results of the above control method including our new proposal described in the next section.

3 Proposed Playout Buffer Algorithm to Maximize the Perceived Quality

In our previous work [6], we have demonstrated that **Loss-Control** can control the packet loss ratio as users like. However, there is a high possibility that the delay and other network parameters (type of codecs, access lines) in addition to the PLR would affect the perceived quality. The end users can choose the preferable playout quality, but there are still many configurable parameters left to the end users. It is hence necessary to introduce a simpler index for the end users, which directly indicates the perceived quality in multimedia communications. Today, many metrics to express the playout quality are proposed and evaluated. The subjective metrics that we adopt in this paper are more user-friendly because it is based on scores made by users according to listening and/or watching the played 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 relations between the MOS and network parameters shown in [7] into 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 to apply above MOS-relative function. Numerical comparisons are shown in the next section.

3.1 Effects of Packet Loss Ratio and Delay on MOS

To clear the relation between the MOS value and network parameters, we pick out the data from [7], which show the effects of network parameters on MOS. We show one referred data in Figure 1. Each plot shows the relation between MOS and end-to-end one-way delay in given loss ratio. From the model of one-way delay distribution clarified in our previous work [6], we can get the feasible combinations of playout delay and the packet loss ratio to maximize the MOS index. We describe our modeling method in the next subsection.

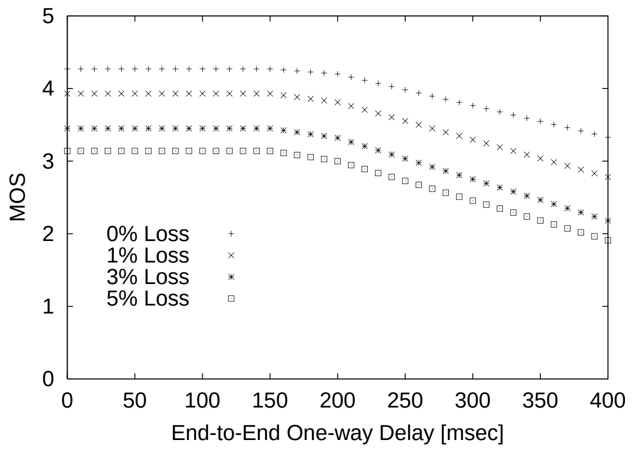


Figure 1: Effects of PLR and Delay (Encoder: G.711)

3.2 Modeling Methods of MOS Functions

The first step of our modeling is to formulate relations among MOS, packet loss ratio, and playout delay approximately. That is, we plot MOS curves shown in Figure 1 with mathematical notations given by our modeling method. Of course, our formulas depends on the relation shown in Figure 1. However, our modeling approach is also applicable to another result of relation.

From Figure 1, we can have the following assumptions.

- Four curves shown in Figure 1 are in parallel. It means that the packet loss ratio and one-way delay, and hence the playout delay, affect MOS independently. From this assumption, we can consider the effect of packet loss ratio and one-way delay on modeling MOS separately.
- The degree of degradation in MOS values is proportional to the packet loss ratio, and it does not depend on the playout delay.

Under above-mentioned assumptions, we can obtain the MOS function $M(p, d)$ for given packet loss ratio p and the 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 given playout delay d with a three-dimensional polynomial approximation, where the packet loss ratio p is assumed to be zero (shown in the cross point in Figure 1).

Parameters 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 get the MOS curve by sliding inversely in the horizontal direction. By applying the second assumption described above, the degree of degradation of MOS values is proportional to the packet loss ratio. We calculate the parameter of the function by the least linear square method. The MOS function $M(p)$ for given packet loss ratio p is hence 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 $M(p, d)$ for given p and d by combining Eqs. (7) and (8), i.e.,

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

In Figure 2 we add the solid curves by Eq. (9) to Figure 1, and we can observe our approximate modeling provides an agreement with the original ones.

In the 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) packet loss ratio caused by packet drops within the network (referred to as p_n), and (2) ratio of late arriving packets exceeding playout threshold (p_d). That is, $p = p_n + p_d$. From Eq. (5) in Section 2, we have a relation between p_d and d as follows;

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

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

$$M(p_n, d) = 4.10 - 0.195\left(p_n + 100\left(\frac{k}{d}\right)^{\alpha}\right)$$

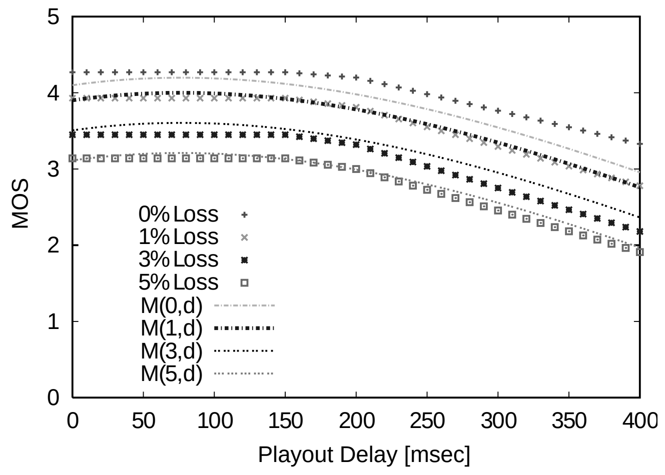


Figure 2: Results of Modeling MOS values and Network Parameters (Encoder: G.711)

$$+2.64 \times 10^{-3} d - 1.86 \times 10^{-5} d^2$$

$$+1.22 \times 10^{-8} d^3. \tag{11}$$

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

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

We now examine $Q(d)$. If $d = 0$, all packets are treated as packet loss, and no packet is played. Thus, we set $Q(0) = 0$. As the playout delay is increased, $Q(d)$ takes smaller value. However, when the playout delay is too large, $Q(d)$ is again degraded due to the large delay for playing. Therefore, there exists an optimum point of d that provides the maximum value of $Q(d)$. Figure 3 shows the 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 from measured data, respectively. Calculating the optimal d is realized by the *false position method* [10] utilizing the differential equation of $Q(d)$. Because $Q(d)$ is the convex function, we can determine the optimal d from the x-intercept of the differential equation of $Q(d)$ by the false position method.

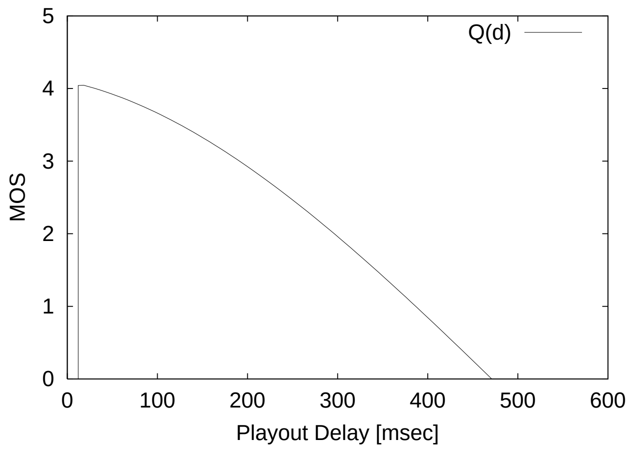


Figure 3: MOS Function $Q(d)$ (Encoder: G.711)

3.3 Modified Playout Buffer Algorithm for Enhancing MOS Index

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

1. Measure the transmission delays of arrived packets
2. Calculate the parameter of Pareto distribution (α, k) by the MLE method (see our prior work [6])
3. Assign the value of (α, k) into the MOS function $Q(d)$
4. Obtain the optimal value of d, maximizing $Q(d)$, by utilizeing the false position method applied 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 enhanced MOS-based playout buffer algorithm (**E-MOS**).

Table 1: 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
p "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

4 Evaluation of Playout Buffer Algorithm

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

4.1 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. Second, in our simulation, the 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 measured delays. Then, we check whether the delay of the next packet is smaller than the estimated playout delay or not. If the delay is larger than the estimated playout delay, the packet is treated to be lost. After tracing all measured delays, average playout delay and packet loss ratio are computed as the output.

4.2 Simulation Results

Now, we evaluate the performance of **E-MOS** by simulations. Table 1 compares packet loss ratios (PLRs), mean values of the playout delays, and MOS evaluated by simulation in the following three cases; The first case is "dynamic", in which the values of 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 the dial-up line. The second is a "moderate" case, in which there are a several number of spikes. We used the G.711 encoder on ADSL and the delays were measured at 1 PM. The last case is "quiet", where no dynamic change of delays were observed. These delays were sent by the G.723.1 encoder at 2 PM and delivered to the receiver through LAN. In **Loss-Control**, we used 95, 99, and 99.9% as the target values. The MOS values shown in the last column of Table 1 are evaluated by Eq. (9) from the PLR and playout delay. The maximum value of MOS among all PBAs is shown in bold.

Results in Table 1 indicate that **E-MOS** can provide the highest perceived quality for users in any network conditions. Looking at the playout delays and PLRs of **E-MOS**, we can find that **E-MOS** has a tendency to minimize the PLR when the one-way delays are small ("moderate" and "quite" cases in Table 1). From Figure 1, we can observe that the effect of introducing the playout delay is quite limited when the playout delay is small (less than 200 msec). In this region, it is effective to prevent the packet loss by lengthening the playout delay. However, as the one-way delay becomes large, **E-MOS** tries to intentionally bear the increasing PLR for reducing the playout delay. It is a good solution for improving the users' perceived quality. Other PBAs have their own ground. For example, **Window** gives a good result in the "dynamic" case, but it is worse than **F-Exp-Avg** in other cases. However, **F-Exp-Avg** provides quite poor performance in the "dynamic" condition. **Exp-Avg** and **Loss-Control (99.9%)** give a passable performance in all cases. However, these methods cannot attain the improvement of the perceived quality as **E-MOS**. Furthermore, the **Loss-Control** method has a disadvantage that it tries to shorten the playout delay and forces to abandon packets even if the playout delay is enough short (less than 200 msec). Thus, **Loss-Control** is not suitable in low packet transmission delay environments.

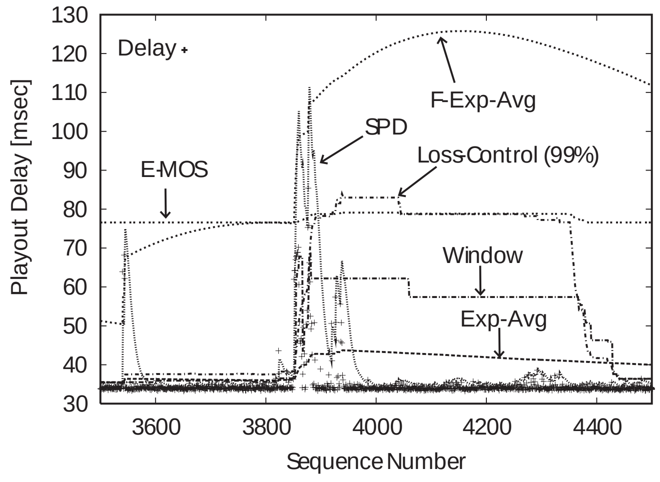
Figure 4 shows the time-dependent behavior of the playout delay, PLR, and MOS for PBAs. Here, the target value of **Loss-Control** is set to 99%. The playout delays of **E-MOS** are larger than the other algorithms except **F-Exp-Avg**, where the one-way delays are small. From this figure, we find that **E-MOS** intends to minimize the PLR when the one-way delay is less than 200 msec. On the other hand, in the condition that the one-way delay is over 200 msec, **E-MOS** tries to increase the PLR for reducing the playout delay in order to enhance the MOS. That is, **E-MOS** can achieve a good balance of the playout delay and PLR based on Eq. (12).

4.3 Evaluation through Implementation Experiments

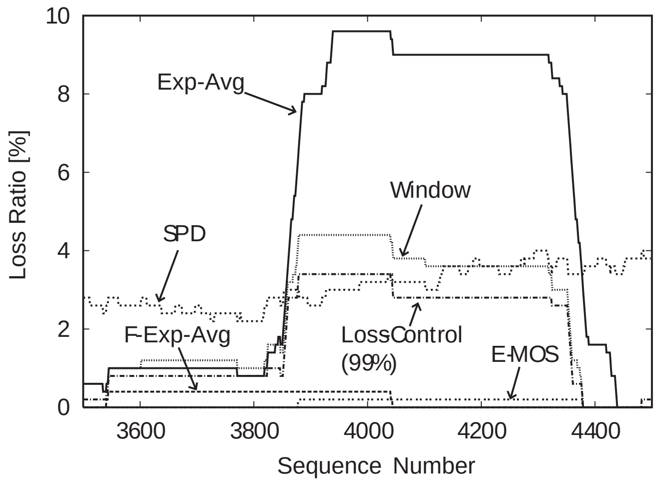
We developed a streaming client in which our PBA is implemented, and verified the applicability of our algorithm by running the application. More specifically, we implemented our PBA as an input plug-in of `Winamp` [11], which is one of major frontends in real-time applications today.

We placed the streaming server at Osaka University, which sends audio packets generated by the G.711 or G.728 encoder (the size of the packet and transmission interval are 160 byte and 20 msec for G.711 and 40 byte, 20 msec for G.728, respectively). Packets are transmitted via the Internet and transferred to our developed client. In the client, the smoothing buffer is adjusted based on the playout delay calculated by our PBA (**E-MOS**). Arrived packets are stored into the buffer and then the client starts playing after the playout interval. Figure 5 shows the operation window of our client.

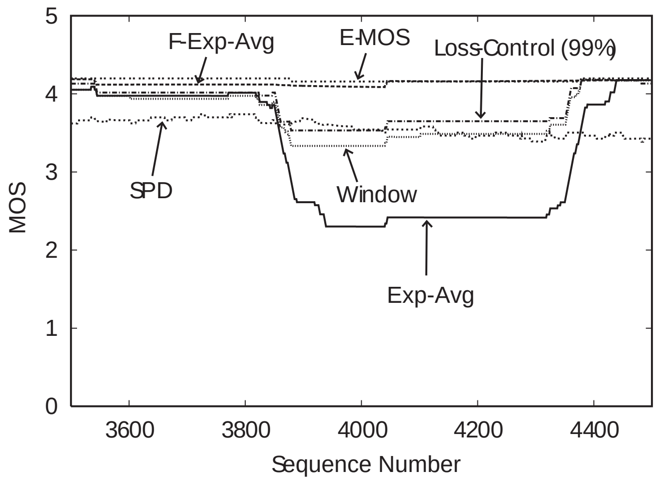
Our implementation experiments include (1) to check whether our PBA tries to maximize the MOS, (2) to



(a) Comparison of Playout Delay

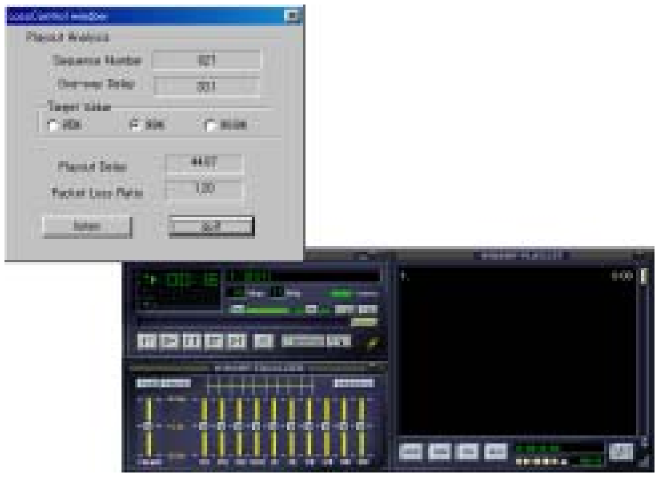


(b) Comparison of PLR



(c) Comparison of MOS

Figure 4: Performance Evaluation of Each PBA (“moderate” case)


Playout Analysis

Sequence Number 921

Overlay Delay 351

Target Value

90% 99% 99.9%

Playout Delay 44.67

Packet Loss Ratio 1.20

Figure 5: Operation Window of Client†

verify whether the computational overhead of calculating the playout delays is enough small to operate our PBA in *real-time*.

The platform was Microsoft Windows 98 operating system on Intel Pentium III 750 MHz CPU. In this case, the computation overhead was about 0.02 msec for each packet arrival; 0.1% of packet transmission interval in G.711, which is sufficiently small overhead. Note that we also confirmed that the audio playing is not interrupted by any other factors except packet losses.

[†] ©Nullsoft Inc. 2002

5 Concluding Remarks

In this paper, we have attended to the perceived quality of streaming applications and then, modify the proposed algorithm so that perceived quality may become maximum. Simulation and implementation experiments have shown that the modified algorithm performs the highest quality in all of PBAs

For future research topics, it is necessary to improve the accuracy of our model for representing the delay distributions. To achieve it, it might be useful to test another heavy-tailed probability functions as the model of delay distributions. Moreover, though no serious problem occurs at the client of **E-MOS**, less CPU load would be more comfortable for users. The more effective calculation method for **E-MOS** is necessary.

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

[8] 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.

[9] 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.

[10] 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.

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