# ADAPTIVE PLAYOUT SCHEDULING USING TIME-SCALE MODIFICATION IN PACKET VOICE COMMUNICATIONS

*Yi J. Liang, Nikolaus Färber, Bernd Girod*

Information Systems Laboratory, Department of Electrical Engineering
Stanford University, Stanford, CA 94305
{yiliang, nfaerber, bgirod}@stanford.edu

## ABSTRACT

A new receiver-based playout scheduling scheme is proposed, which estimates the network delay from past statistics and adaptively adjusts the playout time of the voice packets. In contrast to previous work, the adjustment is not only performed in between talkspurts, but also within the talkspurts in a highly dynamic way. Proper reconstruction of continuous output speech is achieved by scaling individual voice packets using a time-scale modification technique which modifies the rate of playout while preserving voice pitch. Subjective listening tests show that this operation does not impair audio quality. Simulation results based on Internet measurement indicate that buffering delay and loss rate can be significantly reduced by adaptive scheduling.

## 1. INTRODUCTION

Quality of service (QoS) has been one of the major concerns in the context of voice communication over packet networks such as the Internet. Legacy networks and today's Internet protocol, however, only provide services on a best-effort basis without any promise of quality. Transmission delay, delay variation (also known as jitter) and loss have been the major challenges for real-time voice communications. Considerable efforts have been made in different layers of current communication systems to decrease the delay, smooth the jitter and recover the loss.

One important functionality to be implemented on the application layer is the playout scheduling of the voice packets. Playout scheduling is desired at the receiving end in order to continuously play out audio despite delay variations on the network, and the scheduling algorithm greatly affects the quality of service delivered to the end user. In previous work [1] - [4], a playout buffer is employed at the receiver to absorb the delay jitter before outputting the audio. This introduces additional buffering delay by holding packets until their scheduled playout time. Any packet arriving later than the scheduled deadline has to be discarded, resulting in loss. Obviously, there exists a tradeoff between delay and loss. Scheduling a later deadline increases the possibility of playing out more packets and results in lower loss rate, but at the cost of higher buffering delay. Vice versa, it is difficult to decrease the buffering delay without boosting the loss rate.

In this paper we propose a new playout scheduling scheme to improve this tradeoff. In this scheme, we adaptively adjust the playout schedule of each individual packet according to the varying network condition, even during voiced periods. The proper reconstruction of continuous output audio is achieved by scaling the voice packets using a time-scale modification technique. By adjusting the playout time in a more dynamic way, we are able to reduce the delay and loss rate significantly and improve the overall performance.

This paper is organized as follows. Section 2 describes the basic principle of the new algorithm. Section 3 describes the scaling of the voice packet using time-scaling techniques. Section 4 presents the results of performance comparison of different algorithms and subjective listening tests, and Section 5 presents our conclusions.

## 2. ADAPTIVE PLAYOUT SCHEDULING

Fig. 1 (a)-(c) illustrate the three basic scheduling schemes that are investigated in this paper. The simplest method, denoted as *Algorithm 1* in the following, uses a fixed playout deadline for all the voice packets in a session, as depicted in Fig. 1 (a). It is not very effective in keeping both delay and loss rate low enough, as the statistics of the network delay keep changing and a fixed playout time does not reflect this variation. Improved playout algorithms were proposed in [1] - [4], which keep monitoring the network delay and adaptively adjust the playout time during unvoiced periods. This is based on the observation that during a typical conversation, the session can be grouped into talkspurts gapped by silence periods. The playout time of a new talkspurt may therefore be adjusted by extending or shortening the silence periods between talkspurts. This approach is denoted *Algorithm 2* in the following and provides some advantage over Algorithm 1 as illustrated in Fig. 1 (b). However, the effectiveness is limited when the talkspurts are long, and the network delay variation is high within talkspurts.

In the new scheduling scheme proposed in this paper, the playout time is not only adjusted in silence periods, but also within talkspurts. Each individual packet may have a different scheduled playout time, which is set according to the varying delay statistics. This method is denoted as *Algorithm 3* and illustrated in Fig. 1 (c). For the same delay trace, the new algorithm is able to effectively mitigate loss by adapting the playout time in a more dynamic and reactive way to the varying network delay.

To describe the dynamic selection of playout times more precisely, we introduce the following notation illustrated in Fig. 2. We denote the time when a packet is sent, received, and played out by $t_s^i$, $t_r^i$ and $t_p^i$ respectively, where $i = 1, 2, \dots N$ denotes the packet sequence number. We assume that the sender transmits packets at a constant *packetization time* $L_0$, i.e., $t_s^{i+1} - t_s^i = L_0 = const.$. The *buffering delay* is then given by $d_b^i = t_p^i - t_r^i$, while the *net-*

This work has been supported by a gift from Netergy Networks, Inc., Santa Clara, CA.

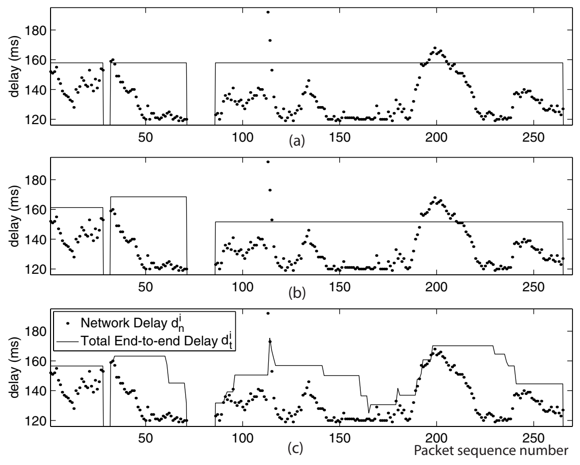

**Fig. 1**. Different playout scheduling schemes (a) fixed playout time (b) between talkspurt adjustment (c) within talkspurt adj.

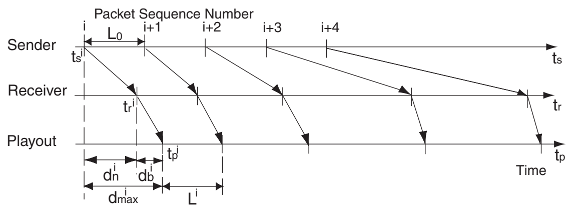

**Fig. 2**. Timing relationship for adaptive scheduling.

*work delay* $d_n^i$ is given by $d_n^i = t_r^i - t_s^i$. The total end-to-end delay $d_t^i$, is the sum of the two quantities above, i.e., $d_t^i = d_n^i + d_b^i$. We also denote the number of packets sent in a stream by $N$, and the set of received packets by $\mathcal{R} = \{i | t_r^i < \infty\}$.

The task of a particular scheduling scheme is to set the maximum allowable total delay $d_{max}^i$ (playout deadline) for each packet. Note that for Algorithms 1 and 2, $d_{max}^i = d_{max} = const.$ for all $i$ belonging to the session or the same talkspurt respectively. Only for Algorithm 3, the adaptation is actually performed on a packet by packet basis. As a result, the length of packets that are played out may differ for each packet, i.e.,

$$t_p^{i+1} - t_p^i = L^i \; . \tag{1}$$

where $L^i$ is the achieved length (in time) of audio packet $i$. The required scaling of voice packets is discussed in Section 3.

When evaluating different scheduling schemes we are primarily interested in two quantities. The first one is the average buffering delay, which is given by

$$d_b = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} (d_{max}^i - d_n^i) \; , \tag{2}$$

where $\mathcal{P} = \{i | t_p^i > t_r^i\}$ is the set of played packets. The second quantity is the associated late loss rate, given by

$$\varepsilon_l = (|\mathcal{R}| - |\mathcal{P}|)/N \; . \tag{3}$$

These two metrics also reflect the above mentioned tradeoff between loss and delay and are used to compare the performance of different playout scheduling algorithms in Section 4.1. For completeness we also define the link loss rate as $\varepsilon_n = (N - |\mathcal{R}|)/N$. The total loss rate is the sum of these two quantities, i.e., $\varepsilon = \varepsilon_n + \varepsilon_l$.

The basic operation of the playout process is as follows. When a new packet $i$ arrives, its network delay $d_n^i$ is obtained from the RTP header it carries. Delay $d_n^i$, together with past delays, is taken into consideration to estimate delay $d_n^{i+1}$ and playout time $t_p^{i+1}$ of the next incoming packet. The current packet $i$ is then scaled immediately, if necessary, according to $L^i$ calculated by (1). If the delay of the next packet is correctly estimated, the next packet should arrive before the last sample of the current packet is played. Usually packets are scaled to either retard the speech when the network delay is increasing, or speed up the speech when it is decreasing. It is important to note that this scheme does not introduce any additional delay other than processing delay.

It can be observed from (2) that for Algorithm 3, if the playout deadline $d_{max}^i$ is chosen close enough to $d_n^i$, the buffering delay can be reduced. At time $i$, we set $d_{max}^{i+1}$ and hence the playout time $t_p^{i+1}$ for the next packet based on past delay statistics. The data and histogram of the past $w$ packet delays are collected and kept, and the histogram of past delays accumulated at time $i$ is denoted by $h^i(d_n)$, with the superscript $i$ indicating the time when the histogram is updated. According to the user-specified loss rate $\hat{\varepsilon}_l$, $d_{max}^{i+1}$ is set to the smallest value satisfying

$$\hat{\varepsilon}_l \leq \sum_{d_n = d_{max}^{i+1}}^{\infty} h^i(d_n) / \sum_{d_n=0}^{\infty} h^i(d_n) \; . \tag{4}$$

The expected buffering delay $\hat{d}_b^{i+1}$, given that packet $i+1$ is played out (i.e., $d_n^{i+1} < d_{max}^{i+1}$), is

$$\hat{d}_b^{i+1} = \sum_{d_n=0}^{d_{max}^{i+1}} (d_{max}^{i+1} - d_n) h^i(d_n) / \sum_{d_n=0}^{d_{max}^{i+1}} h^i(d_n) \; . \tag{5}$$

Stored past delays are updated continuously, with the oldest values being discarded and the most recent added. A good feature about the histogram-based estimation is that the resulting performance can be directly associated with QoS parameters such as the loss rate. The user has the option in setting the parameter. In practice, due to the stringent budget of the end-to-end delay and the good performance of available loss concealment techniques, we usually tend to trade the loss rate for even lower delay in order to achieve better overall performance.

## 3. SCALING OF VOICE PACKETS

The scaling of voice packets is realized by *time-scale modification* based on the *Waveform Similarity Overlap-Add* (WSOLA) algorithm, which is an interpolation-based method operating entirely in the time domain. This technique was first used in [5] to scale long audio blocks, and modified and improved in [6] and [7] for loss concealment by expanding a block of several packets, where a delay of 2–3 packet times is introduced due to the multi-packet operation. Here we have tailored the WSOLA algorithm and improved it to work on only *one* packet, so that no algorithm delay will be introduced during the packet scaling - playout process.

To scale an audio packet, we first define a *template segment* of constant length in the input, and then search for the *similar segment* which has the maximum similarity to the template segment.

The start of the similar segment is searched in a *search region*, as is shown in Fig. 3. When working on a single packet, it is more difficult to locate a similar segment in the input through correlation calculation, since the realignment of the similar segments must be done in units of pitch periods and there are fewer pitch periods in one short packet. To work out this problem, we modified the WSOLA algorithm to decrease the segment length for correlation calculation, and to position the first template segment at the beginning of the input packet, as shown in Fig. 3(a). For expanding short packets, we also move the search region for the first similar segment to the prior packet in order to have a larger range to look for similar waveforms, as is suggested by [6]. Although the prior packet has already been played out at the time of scaling, similar waveforms can still be extracted from it to construct new output without delaying the prior packet. Once the similar segment is found, it is weighted by a rising window and the template segment weighted by a symmetric falling window. The similar segment followed by the rest of the samples in the packet is then shifted and superimposed with the template segment to generate the output. The resulting output is longer than the input due to the relative position of the similar segment found and the shift of the similar segment, as is shown in Fig. 3(a). The amount of expansion depends on the position and size of search region defined.

In Fig. 3(a), we can observe from the output waveform that one extra pitch is created and added as a result of realignment and superposition of the extracted segments from the input. However, the extra pitch is not just a simple replication of any pitch from the input, but the interpolation of several pitches instead. This explains the reason why the sound quality by time-scale modification is better than that by pitch repetition (described in [8]). Same is true for compressing a packet, where the information carried by a chopped pitch is preserved and shared by the remaining ones. However, the single-packet operation described above has the same advantage as pitch repetition in terms of no added delay.

Packet compression is done in a similar way, with the only difference being the search region position due to the smaller output length desired. Comparing the input and output waveforms in Fig. 3 (a) or (b), it becomes obvious that the operation does not modify the pitch period, or tone, of the input speech. The only thing changed is the packet length and hence the rate of speech.

An important feature of the operation illustrated in Fig. 3 is that the beginning and ending samples of the output are kept the same as those in the input, so that the output and input have the same interface for concatenation. Modified packets can be sent to the output queue one after another, back to back, with no need for merging. This kind of "black box" operation is independent for each single packet and is very flexible, which is ideal for adaptive playout time adjustment.

However, as one notices from Fig. 3, the increment and decrement of the packet size has to be at least one pitch period to preserve the tone of speech and the same packet interface for concatenation. This discretization of the scaling ratio is in contrast to the desired playout time and packet length being continuous quantities. For this reason, we define *expansion* and *compression thresholds* for packet scaling. Only when the desired playout time advances the current playout time by more than the compression threshold, do we actually compress a packet to speed up the playout. This threshold is usually greater than a typical pitch period of the present speech. The same is true for stretching a packet, except that the two thresholds are asymmetric. To prevent unnecessary late loss, we do packet compression conservatively enough to

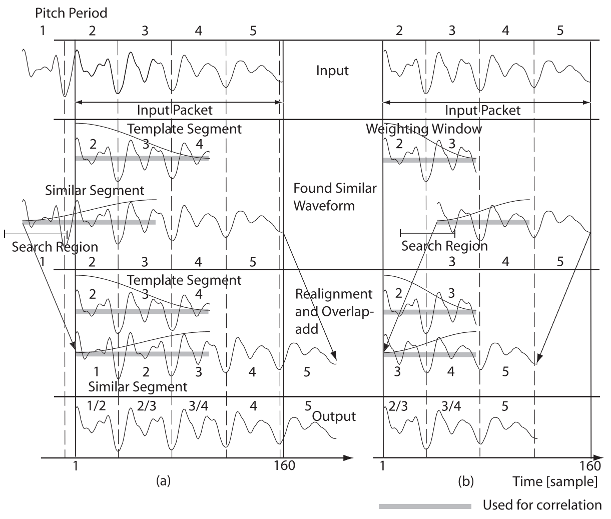

**Fig. 3**. Scaling of a voice packet using time-scale modification (a) extension (b) compression.

avoid dropping the playout time below what we targeted. On the other hand, by defining smaller expansion thresholds, we stretch the packet and slow down the playout more readily in order to adapt to sudden increase of the network delay faster and thus to avoid any late loss. This kind of asymmetry can be clearly observed in Fig. 1(c). The use of thresholds also introduces hysteresis into the algorithm and helps to smooth out the playout jitter.

## 4. PERFORMANCE COMPARISON AND RESULTS

### 4.1. Performance Comparison

We compare the performance of the three playout scheduling schemes described in Section 2, i.e., Algorithms 1-3. We collected packet delay traces over the Internet by transmitting RTP streams from a host at Stanford University to remote hosts in 1) Chicago, 2) Germany, 3) MIT, and 4) China, referred to as Traces 1-4 respectively. The measured data represent one-way delay of UDP packets with payload size of 160 bytes, reflecting 20 ms G.711 coded voice packet using 8-bit quantization and 8 KHz sampling rate. Each trace covers 9000 packets. The playout of the voice packets is simulated over each trace using the three algorithms under comparison. The continuous curves with different loss rate and buffering delay in Fig. 4 are obtained by varying the control parameters of each particular algorithm, e.g., the user-specified loss rate determining the playout deadline in Algorithm 3.

Fig. 4 shows the total loss rate (equivalent to the late loss rate)/link loss rate vs. average buffering delay using different algorithms for the four traces. If targeting a total loss rate of 5% for Trace 1, the average buffering delay is reduced by 40.0 ms when using Algorithm 3 instead of 1. Comparing Algorithm 3 with 2 the gain is still 31.5 ms. If allowing the same buffering delay for different algorithms, Algorithm 3 results in the lowest loss rate. For Trace 1, if the same 40 ms average buffering time is allowed, the total loss rate resulting from Algorithm 3 is more than 10% lower than that from Algorithms 1 and 2. Similar reductions in buffering delay and loss rate are also obtained in Traces 2-4.

The performance gain of Algorithm 3 over Algorithms 1 and 2

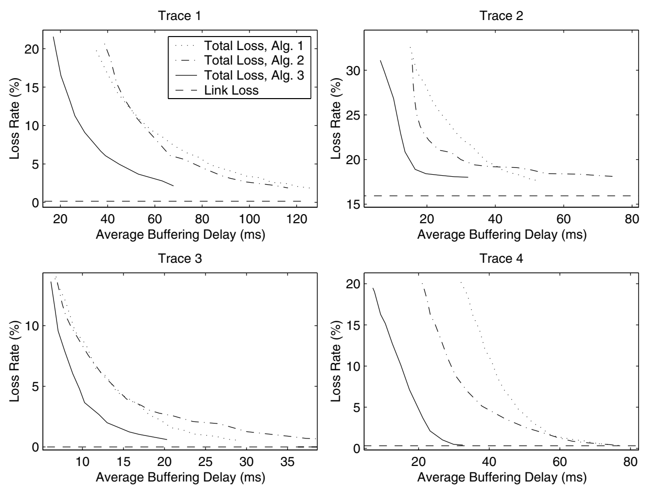

**Fig. 4**. Performance of three playout scheduling algorithms.

**Table 1**. Subjective test results of packet scaling.

| Network Cond. | STD of Network Delay (ms) | STD of End-to-end Delay (ms) | % of Packet Scaled | Score |
|---|---|---|---|---|
| 1 | 19.6 | 7.5 | 17.8 | 4.7 |
| 2 | 20.9 | 10.5 | 18.4 | 4.5 |
| 3 | 65.0 | 28.2 | 24.1 | 4.6 |

differs from trace to trace though. The performance of Algorithm 3 depends on the accurate prediction of the next delay based on past delays, and the gain over the other algorithms also depends on the jitter statistics. For Trace 3, the link between Stanford and MIT has high throughput, and the jitter is the mildest among all traces, which explains the lowest gain.

### 4.2. Subjective Listening Test Results

As seen from previous sections, the superior performance of Algorithm 3 depends on adaptive playout time adjustment, which is enabled by packet scaling. We performed subjective listening tests on scaled audio following the *degradation category rating* (DCR) method described in Annex D of Recommendation P.800 [9]. In DCR the speech samples are presented in pairs, in the format of "original sample - processed sample". The listeners are asked to rate the processed sample on a 5-point scale with grades corresponding to *5-degradation inaudible*, *4-audible but not annoying*, *3-slightly annoying*, *2-annoying*, *1-very annoying*.

Three short traces are extracted from the delay data collected, which represent medium jitter, high jitter, and extremely high jitter situations, respectively. This yields three processing conditions listed in Table 1. We simulated playing out six speech samples under each condition using Algorithm 3 with packet scaling to meet the timing requirement of adaptive scheduling, and generated the samples to be rated. The scaling ratio (output length/input length) of the voice packet is between 0.35 and 2.30. In order to focus on testing the scaling effect, the processed samples do not include any packet loss. The scores are the averages from eighteen listeners.

As can be seen from Table 1, the degradation due to audio scaling is almost inaudible, even for extreme cases. One reason for this finding is that packets actually do not have to be scaled very frequently as shown in column 4 of Table 1. Even for Condition 3, fewer than one fourth of the packets were actually scaled to satisfy the timing requirement of adaptive scheduling.

Also listed in Table 1 are the standard deviation (STD) of the network delay, which reflects the jitter characteristics for each condition, as well as the standard deviation of the total end-to-end delay, which characterizes the variation of the playout rate, or playout jitter, within talkspurts. For Algorithm 1, the latter quantity would be zero. For Algorithm 3, the playout is not necessarily jitter-free. However the jitter is significantly smoothed by the scheduling algorithm, as is observed from columns 2 and 3 in the table.

## 5. CONCLUSIONS

In this paper we presented a new receiver-based playout scheduling scheme which improves the end-to-end quality for voice communication over packet networks. The proposed algorithm effectively reduces buffering delay and loss rate by adapting the playout schedule to the varying network delay in a highly dynamic way. Simulation results based on Internet measurement have shown significant reductions in average buffering delay and loss rate produced by the proposed playout scheme. Subjective listening test results have also substantiated the good quality of the scaled speech.

## 6. REFERENCES

[1] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proc. IEEE INFOCOM '94*, June 1994, vol. 2, pp. 680–688.

[2] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: Performance bounds and algorithms," *Multimedia Systems*, vol. 6, no. 1, pp. 17–28, Jan. 1998.

[3] J. Pinto and K. J. Christensen, "An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods," in *Proc. 24th Conference on Local Computer Networks*, Oct. 1999, pp. 224–231.

[4] P. DeLeon and C.J. Sreenan, "An adaptive predictor for media playout buffering," in *Proc. ICASSP99*, Mar. 1999, vol. 6, pp. 3097–3100.

[5] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *Proc. ICASSP 93*, Apr. 1993, pp. 554–557.

[6] A. Stenger, K. Ben Younes, R. Reng, and B. Girod, "A new error concealment technique for audio transmission with packet loss," in *Proc. European Signal Processing Conf.*, Sept. 1996.

[7] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod, "A new technique for audio packet loss concealment," in *IEEE GLOBECOM*, Nov. 1996, pp. 48–52.

[8] D. J. Goodman *et al.*, "Waveform substitution techniques for recovering missing speech segments in packet voice communications," *IEEE Trans. ASSP*, vol. ASSP-34, no. 6, pp. 1440–1448, Dec. 1986.

[9] ITU-T Recommendation P.800, *Methods for Subjective Determination of Transmission Quality*, Aug. 1996.