Adaptive Playout Scheduling and Loss Concealment for Voice Communication Over IP Networks

Yi J. Liang, *Member, IEEE*, Nikolaus Färber, *Member, IEEE*, and Bernd Girod, *Fellow, IEEE*

***Abstract*—The quality of service limitation of today's Internet is a major challenge for real-time voice communications. Excessive delay, packet loss, and high delay jitter all impair the communication quality. A new receiver-based playout scheduling scheme is proposed to improve the tradeoff between buffering delay and late loss for real-time voice communication over IP networks. In this scheme the network delay is estimated from past statistics and the playout time of the voice packets is adaptively adjusted. In contrast to previous work, the adjustment is not only performed between talkspurts, but also within talkspurts in a highly dynamic way. Proper reconstruction of continuous playout speech is achieved by scaling individual voice packets using a time-scale modification technique based on the Waveform Similarity Overlap-Add (WSOLA) algorithm. Results of subjective listening tests show that this operation does not impair audio quality, since the adaptation process requires infrequent scaling of the voice packets and low playout jitter is perceptually tolerable. The same time-scale modification technique is also used to conceal packet loss at very low delay, i.e., one packet time. Simulation results based on Internet measurements show that the tradeoff between buffering delay and late loss can be improved significantly. The overall audio quality is investigated based on subjective listening tests, showing typical gains of 1 on a 5-point scale of the Mean Opinion Score.**

***Index Terms*—Jitter absorption, loss concealment, playout scheduling, time-scale modification, voice over IP.**

I. Introduction

THE unreliable and stateless nature of today's Internet protocol (IP) results in a best-effort service, i.e., packets may be delivered with arbitrary delay or may even be lost. This quality-of-service (QoS) limitation is a major challenge for real-time voice communication over IP networks (VoIP). Since excessive end-to-end delay impairs the interactivity of human conversation, active error control techniques such as retransmission cannot be applied. Therefore, any packet loss directly degrades the quality of the reconstructed speech. Furthermore, delay variation (also known as jitter) obstructs the proper reconstruction of the voice packets in their original sequential and periodic pattern.

Considerable efforts have been made in different layers of current communication systems to reduce the delay, smooth the jitter, and recover the loss. On the application layer, receiver-based, passive methods have the advantage that no cooperation of the sender is required. Furthermore, they can operate independently of the network infrastructure. One important functionality to be implemented at the receiver is the concealment of lost packets, i.e., the recovery of lost information based on the redundancy in neighboring packets. Another functionality that is discussed in detail in the following is the playout scheduling of voice packets.

The common way to control the playout of packets is to employ a playout buffer at the receiver to absorb the delay jitter before the audio is output. When using this *jitter absorption* technique, packets are not played out immediately after reception but held in a buffer until their scheduled playout time (*playout deadline*) arrives. Though this introduces additional delay for packets arriving early, it allows to play packets that arrive with a larger amount of delay. Note that there is a tradeoff between the average time that packets spend in the buffer (*buffering delay*) and the number of packets that have to be dropped because they arrive too late (*late 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 significantly increasing the loss rate. Therefore, packet loss in delay-sensitive applications, such as VoIP, is a result of not only packet being dropped over the network, but also delay jitter, which greatly impairs communication quality. In the experiments presented in Sections VI and VII, we find the maximum delay jitter (the difference between the maximum and the minimum network delay in a particular trace) in different traces collected across the WAN between 39 and 130 ms, depending on the network setup and the link condition.

Previous work mainly focused on improving the tradeoff between delay and loss, while trying to compensate the jitter completely or almost completely within talkspurts [1]–[5]. By setting the same fixed time for all the packets in a talkspurt, the output packets are played in the original, continuous, and periodic pattern, e.g., every 20 ms. Therefore, even though there may be delay jitter on the network, the audio is reconstructed without any *playout jitter*. Some recently proposed schemes [6], [7] apply adaptive scheduling of audio and other types of multimedia, accepting certain amount of playout jitter. However, in these methods, the playout time adjustment is made without regarding the audio signal and it is not addressed how continuous playout of the audio stream can actually be achieved. As a result, the playout jitter that can be tolerated has to be small in order to preserve reasonable audio quality.

In this paper, we propose a new playout scheduling scheme that exploits the increased flexibility of allowing more playout jitter. In this scheme, we adaptively adjust the playout schedule

Manuscript received May 2, 2001; revised April 11, 2002. This work was supported by 8×8, Inc., Santa Clara, CA. The associate editor coordinating the review of this paper and approving it for publication was Dr. Chung-Sheng Li.

The authors are with Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: yiliang@stanfordalumni.org).

Digital Object Identifier 10.1109/TMM.2003.819095

of each individual packet according to the varying network condition, even during talkspurts. The continuous output of high-quality audio is achieved by scaling of the voice packets using a time-scale modification technique. As a result, we can allow a higher amount of playout jitter compared to previous work, which allows us to improve the tradeoff between buffering delay and late loss significantly [8]. This improvement is based on the interesting finding that increased playout jitter is perceptually tolerable if the audio signal is appropriately processed, as demonstrated by subjective listening tests.

This paper is organized as follows. Section II describes the basic idea of adaptive playout and introduces the notation and performance measures used for evaluation. Section III describes how the voice packet can be scaled using time-scaling techniques. In Section IV we show how the network delay is estimated and how the playout schedule is adaptively set accordingly. In Section V, a loss concealment mechanism is proposed that works together with the adaptive playout scheme. Finally, a performance comparison and the subjective quality test results are presented in Sections VI and VII respectively.

II. Fixed versus Adaptive Playout

Fig. 1(a)–(c) illustrate the three basic scheduling schemes that are investigated in this paper. The graphs show the delay of voice packets on the network as dots and the total delay as a solid line. When a later playout time is scheduled, the total delay increases. Packets arriving after the playout deadline, i.e., dots above the line, are lost and have to be concealed. The task of a playout scheduling scheme with respect to Fig. 1 is to lower the solid line (reduce the total delay) as much as possible while minimizing the number of dots above the line (minimize late loss).

The simplest method, denoted as *Algorithm 1,* 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 in practice, because the statistics of the network delay change over time and a fixed playout time does not reflect this variation.

With improved playout algorithms proposed in [1]–[5], the network delay is monitored, and the playout time is adaptively adjusted during silence periods. This is based on the observation that, for a typical conversation the audio stream can be grouped into talkspurts separated by silence periods. The playout time of a new talkspurt may be adjusted by extending or compressing the silence periods. This approach is denoted *Algorithm 2* and provides some advantage over *Algorithm 1* as illustrated in Fig. 1(b). However, the effectiveness is limited when talkspurts are long and network delay variation is high within talkspurts. For example, the silence-dependent method is not able to adapt to the "spike" of high delay within the third talkspurt at packets 113–115. As a result, several packets are lost in a burst causing audible quality degradation.

In the new scheduling scheme proposed in this paper, the playout 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

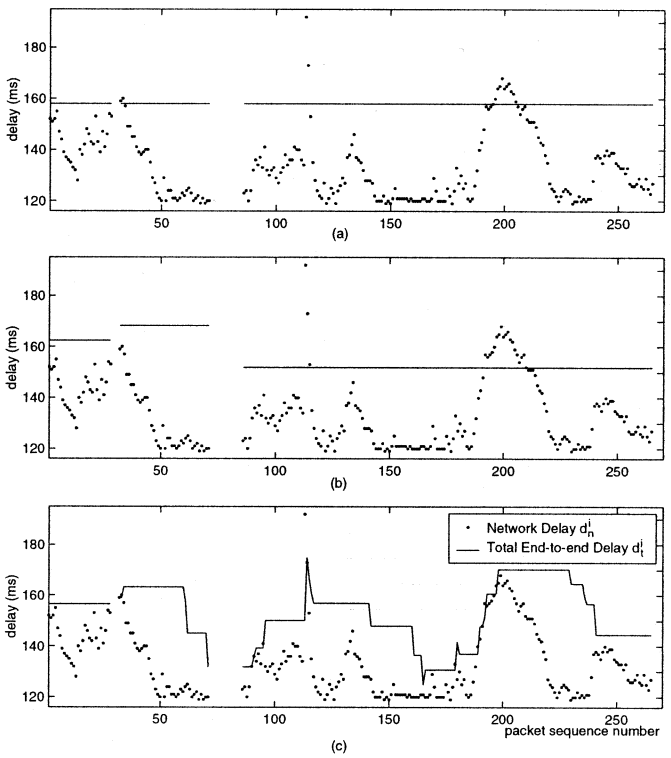


Fig. 1. Different playout scheduling schemes. *Algorithm 1*: fixed playout time (top); *Algorithm 2*: between talkspurt adjustment (middle); *Algorithm 3*: within talkspurt adjustment (bottom). Gaps in solid lines correspond to silence periods between talkspurts.

able to effectively mitigate loss by adapting the playout time in a more dynamic and reactive way. Note that *Algorithm 3* requires the scaling of voice packets to maintain continuous playout and therefore introduces some amount of playout jitter. However, this flexibility allows to reduce the average buffering delay while reducing late loss at the same time. Hence, the tradeoff between buffering delay and late loss is improved.

Even though the Section I already provides an intuitive idea about the task of a playout scheduling scheme and the advantage of adaptive playout, we still need to define an objective performance measure. In the following we therefore introduce the basic notation used in this paper and define the *average buffering delay* and the *late loss rate* as the basic performance measures. For convenience all variables are summarized in Table I.

As 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. The index $i = 1, 2, \ldots N$ denotes the packet sequence number, assuming N packets are sent in the stream. For packet voice, speech is usually processed and packetized into fixed size blocks and outgoing packets are generated periodically at a constant *packetization time* L_0, i.e., $t_s^{i+1} - t_s^i = L_0 = \text{const.}$. The *buffering delay* of packet i is then given by $d_b^i = t_p^i - t_r^i$, while the *network delay* d_n^i is given by $d_n^i = t_r^i - t_s^i$. The total delay d_t^i, is the sum of the two quantities above, i.e., $d_t^i = d_n^i + d_b^i$. Note that this total delay does not include encoding and packetization time (a constant component of the end-to-end delay), since we are mainly interested in packet transmission and speech playout in this work. We use $d_n^i = \infty$ to indicate that packet i is lost during transmission and never reaches the receiver. Hence, the set of received packets is given by $\mathcal{R} = \{i | t_r^i < \infty\}$.

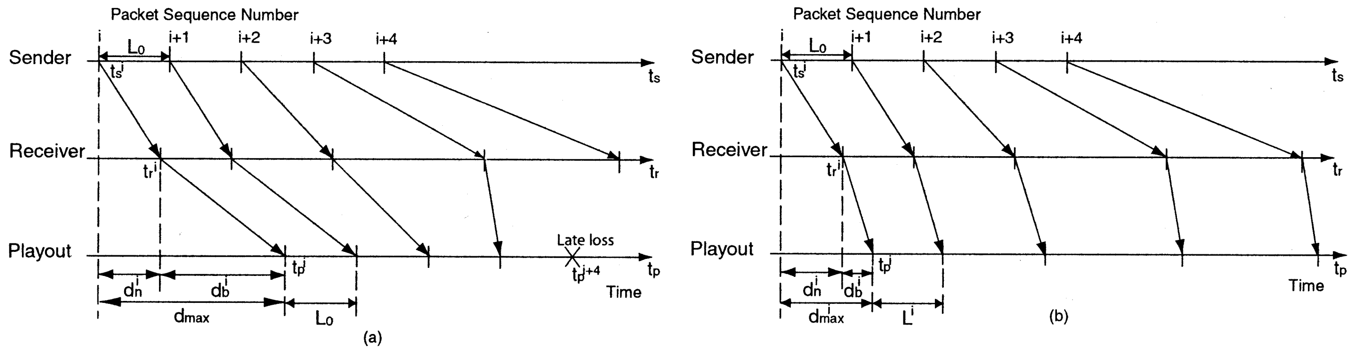


Fig. 2. (a) Fixed and (b) adaptive playout.

TABLE I
BASIC NOTATION

Notation	Description
t_s^i	Time packet i sent
t_r^i	Time packet i received
t_p^i	Time packet i played out
d_n^i	Network delay of packet i
d_b^i	Buffering delay of packet i
d_b	Average buffering delay of a stream
d_t^i	Total delay of packet i
d_{max}	Fixed playout deadline
d_{max}^i	Playout deadline of packet i
D^k	Sorted order statistics of $\{d_n^i\}$
ε_n	Link loss rate
ε_l	Late loss rate
$\hat{\varepsilon}_l$	User-defined late loss rate
ε_b	Burst loss rate
ε	Total loss rate
$\mathcal{R}$	Set of received packets
$\mathcal{P}$	Set of played packets
$\mathcal{B}$	Set of packets lost consecutively
N	Number of packets in a stream
L_0	Sender packetization time
L^i	Actual length of scaled packet i
$\hat{L}^i$	Target length of packet i

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, $d_{\max}^i = d_{\max} = \text{const.}$ for all i belonging to the session or the same talkspurt respectively. Therefore, the playout of packets at the receiver is fixed as illustrated in Fig. 2(a), and the length of output packets is constant, i.e., $L_0 = t_p^{i+1} - t_p^i$. 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.,

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

where L^i is the achieved length (in time) of audio packet i. The resulting adaptive playout is illustrated in Fig. 2(b). The required scaling of voice packets is a major contribution of this work and is discussed in Section III.

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

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

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

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

These two metrics also reflect the above mentioned tradeoff between loss and delay and are used below to compare the performance of different playout scheduling algorithms.

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., $\varepsilon = \varepsilon_n + \varepsilon_l$. Finally, we introduce the *burst loss rate*, denoted by ε_b, to quantify the burstiness of the loss. Burst losses are considered separately because they are more difficult to conceal and impair sound quality more severely. Defining the set of packets with two consecutive losses as $\mathcal{B} = \left\{i|t_p^i < t_r^i, t_p^{i+1} < t_r^{i+1}\right\}$, the burst loss rate is given by $\varepsilon_b = |\mathcal{B}|/N$.

III. Scaling of Voice Packets

As described above, adaptive playout can only be achieved when individual voice packets can be scaled without impairing speech quality. Hence, this functionality is a basic requirement of our work and an appropriate algorithm is described in this section.

The scaling of voice packet 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 used in [9] to scale long audio blocks, and modified and improved in [10] and [11] for loss concealment by expanding a block of several packets. The basic idea of WSOLA is to decompose the input into overlapping segments of equal length, which are then realigned and superimposed to form the output with equal and fixed overlap. The realignment leads to modified output length. For those segments to be added in overlap, their relative positions in the input are found through the search of the maximum correlation between them, so that they have the maximum similarity and the superposition will not cause any discontinuity in the output. Weighting windows are applied to the segments before they are superimposed to generate smooth transitions in the reconstructed output. For speech processing, WSOLA has the advantages of maintaining the pitch period, which results in improved quality compared to resampling.

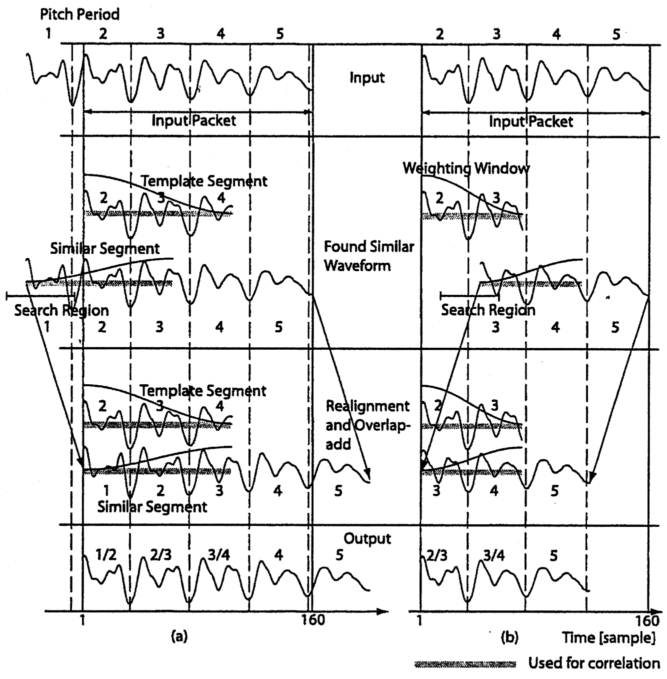


Fig. 3. (a) Extension and (b) compression of single voice packets using time-scale modification.

A. Single Packet WSOLA

In [10] and [11], due to the nature of loss concealment and multi-packet operation, a delay of 2–3 packet times is introduced in expanding the packets. Since the goal of adaptive playout is to cut down delay, lower processing delay is desired in our case. Therefore, we have tailored the WSOLA algorithm and improved it to work on only *one* packet. In other words, an incoming packet can be scaled immediately, without introducing any additional processing delay.

To scale a voice packet, we first select a *template segment* of constant length in the input, and then search for the *similar segment* that exhibits 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, the search for a similar segment is more constrained, 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. For a 20 ms packet, depending on the speaker's gender and voice pitch, there could be fewer than two pitch periods included, which makes it difficult to extract the target segments with similarity. To overcome 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 in [10]. In Fig. 3(a), although the input packet starts in Pitch Period 2, the similar segment is found within Pitch Period 1. Although the prior packet might already be 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 the size of the defined search region.

In Fig. 3, complete pitch periods in the waveform are separated by vertical dashed lines and marked with sequential numbers. For example in Fig. 3(a), we can observe from the output waveform that one extra pitch period is created and added as a result of realignment and superposition of the extracted segments from the input. However, the extra pitch period is not just a simple replication of any pitch period from the input, but the interpolation of several pitch periods instead. For the output in Fig. 3(a), the first three pitch periods are the weighted superposition of Pitch Periods 1/2, 2/3, and 3/4, respectively. This explains why the sound quality using time-scale modification is better than that of pitch repetition (described in [12], [13]). The same is true for compressing a packet, where the information carried by a chopped pitch period is preserved and distributed among the remaining ones. However, the single-packet operation described above has the same advantage as pitch repetition in terms of no added delay.

The operations of searching for a similar segment and extending the packet by multiple pitch periods, as described above, constitute one *iteration* of our scheme. If the output speech has not reached the desired length after such operations, additional iterations are performed. In a subsequent iteration, a new template segment of the same length is defined that immediately follows the template in the last iteration. All the defined template segments and the remaining samples following the last template in the input should cover the entire output with the target length. The total number of defined template segments, and hence the number of iterations used here is

$$\left\lfloor \frac{\hat{L}^i}{W} \right\rfloor - 1 \tag{4}$$

where $\lfloor x \rfloor$ represents the greatest integer number that is smaller than or equal to x, $\hat{L}^i$ is the target length of the output, and W is the length of a segment (either template segment or similar segment).

Packet compression is done in a similar way, as depicted in Fig. 3(b). The only difference is that the search region for the similar segment should not be defined in the prior packet in order to generate an output shorter in length. A successful packet compression requires that a packet contains more than one pitch, which limits the minimum length of the packet that can be compressed. However, a common packet length, such as 20 ms, is usually sufficient since pitch values below 100 Hz are not very frequent in speech signals. If, for some cases, packet compression cannot be performed, it is uncritical to delay the compression to later packets, which will be further discussed in the Section III-B.

Comparing the input and output waveforms in Fig. 3, it becomes obvious that the operation preserves the pitch frequency of the input speech. Only the packet length and hence the rate of speech are altered. Subjective listening tests presented in Section VII show that infrequent scaling of the packets does not degrade the speech quality, even if the scaling ratio is occasionally high. Note that the scheme is entirely voice codec independent. If any kind of codec should be used, the operations can be applied on the PCM output of the decoder.

One advantage of working with a short packet is that the input is divided into fewer template segments so that typically only one or two iterations will yield the output with the desired length. Another important feature of the algorithm apparent in Fig. 3 is that the beginning and the end of each packet are not altered. As a result, when concatenating modified packets, no overlap is needed to obtain smooth transitions. Hence, packets can be modified independently and sent to the output queue back to back. This type of operation is ideally suited for a highly adaptive playout scheduler.

B. Implementation Issues

Since the scaling of packets has to be performed in integer multiples of pitch periods, it is not possible to achieve arbitrary packet lengths and playout times as would be desirable for adaptive playout. In other words, the actual resulting packet length, L^i, after single packet WSOLA can only approximate the required target length, $\hat{L}^i$. For this reason, we define *expansion* and *compression thresholds*. Only if the desired playout time precedes the currently scheduled playout time by more than the compression threshold, we compress a packet to speed up the playout. The compression threshold is usually greater than a typical pitch period. The same strategy is used for the expansion of a packet, except that the two thresholds are asymmetric. To prevent unnecessary late loss, we apply compression conservatively enough to avoid dropping the playout time below what we targeted. On the other hand, smaller expansion thresholds are defined, which might be smaller than a pitch period. In this way, we expand the packet and slow down the playout in order to accommodate sudden increase of the network delay. This asymmetry results in a hysteresis that can be observed in Fig. 1(c), where increased network delay is followed more closely than reduced network delay. The introduced hysteresis also results in smoothed playout jitter.

To avoid extreme scaling, we also define maximum and minimum target packet lengths, denoted by $L_{\max}$ and $L_{\min}$ respectively. In the simulations described in Sections VI and VII, $L_{\max} = 2.3L_0$ and $L_{\min} = 0.3L_0$ are used. However, during silence periods, the amount of adjustment we can make for the playout schedule is not limited by $L_{\max}$ or $L_{\min}$, so that we have more freedom to modify the playout schedule.

The general procedure of playout schedule adjustment is described by the algorithm in Fig. 4. The operation described by line 2, the setting of the playout time, will be discussed in full detail in Section IV.

C. Algorithm Complexity

Due to the real-time nature of the packet scaling operation and low-delay requirement, the algorithm has to be computationally efficient. Hence, we briefly analyze the complexity of single packet WSOLA in this section. Denote the length of a segment in samples by W, and the length of search region in samples by R. In one iteration, the number of operations for correlation calculation is WR multiplications, plus $2W$ multiplications for windowing. For a typical 20 ms packet sampled at 8 kHz, if limiting the maximum scaling ratio to be $L_{\max}/L_0 = 2.3$, there would be at most three iterations in total according to (4). Considering typical values of $W = 80$ and $R = 100$, and three iterations, the maximum complexity of scaling one packet is approximately 24 000 multiplications and 24 000 additions. Based on experiments on a 733 MHz Pentium III machine, this operation requires approximately 0.35 ms. In practice, scaling by $L_{\max}/L_0$ is carried out infrequently, and the average load will be significantly lower than the peak load estimated above.

```
1  Receive packet $i$;
2  Estimate and set the playout time for packet $i+1$, $\hat{t}_p^{i+1}$;
3  Calculate the desired length of packet $i$, $\hat{L}^i = \hat{t}_p^{i+1} - t_p^i$;
4  if $\hat{L}^i - L_0 > expansion\ threshold$
5      Scale packet $i$ with target length $\min(\hat{L}^i, L_{max})$;
6  elseif $\hat{L}^i - L_0 < -compression\ threshold$
7      Scale packet $i$ with target length $\max(\hat{L}^i, L_{min})$;
8  else
9      Keep packet $i$ without modification;
10 endif
11 Output packet $i$ with actual length $L^i$;
12 Update the playout time of packet $i+1$, $t_p^{i+1} = t_p^i + L^i$;
```

Fig. 4. Algorithm for adaptive playout time adjustment with packet scaling.

IV. Setting the Playout Schedule

The basic operation of the playout scheduler is to set the playout time t_p for each packet. Before packet i can be played out we need to decide on the length L^i to perform the required scaling. According to (1), this implicitly sets the playout time of the next packet to $t_p^{i+1} = t_p^i + L^i$. Therefore, in order to play packet i, we need to estimate the arrival and playout time of packet $i+1$, or equivalently, estimate the network delay d_n^{i+1}, which is Step 2 in Fig. 4. If the delay of the next packet is correctly estimated, the next packet should arrive in time and be ready by the end of playback of the current packet. A good estimation of the network delay is therefore an important component for adaptive playout scheduling. Known techniques for delay estimation include linear recursive filtering with stochastic gradient algorithms [1], histogram based approaches [2], [7], normal approximation [14], and event-counting [6]. Here we use the delay of past packets and base our estimation on its order statistics in order to adapt the playout schedule to the network variations in a more reactive way.

A. Order Statistics Based Estimation

In our approach, the delays are collected for a sliding window of the past w packets. A threshold of the total delay for the next packet, $d_{\max}^{i+1}$, is defined according to the user-specified loss rate, $\hat{\varepsilon}_l$. The next packet must arrive before that deadline in order to be played out. The determination of $d_{\max}^{i+1}$ is described in detail as follows.

The network delay of past w packets recorded is d_n^{i-w+1}, $d_n^{i-w+2}, \ldots, d_n^i$. Its order statistics, or the sorted version of $d_n^{i-w+1} d_n^{i-w+2}, \ldots, d_n^i$ are denoted as $D^1, D^2, \ldots, D^w$, where

$$D^1 \leq D^2 \leq \cdots \leq D^w. \tag{5}$$

The probability that network delay d_n is no greater than the r-th order statistic D^r is

$$F(D^r) = P(d_n \leq D^r), \quad r = 1, 2, \ldots, w.$$

In [15], it is shown that

$$\mathcal{E}(F(D^r)) = \frac{r}{w+1}, \quad r = 1, 2, \ldots, w \tag{6}$$

which is the expected probability that a packet with the same delay statistics can be received by D^r.

In our application, we extend (5) by adding an estimate of the lowest possible delay $D^0 = \max(D^1 - 2s_{d_n}, 0)$, and the maximum delay

$$D^{w+1} = D^w + 2s_{d_n} \tag{7}$$

where s_{d_n} is the standard deviation of the sample $d_n^{i-w+1}, d_n^{i-w+2}, \ldots, d_n^i$, such that we obtain the extended order statistics

$$\begin{aligned} D^0 \leq & D^1 \leq \cdots \leq D^w \leq D^{w+1}, \text{ and} \\ \mathcal{E}(F(D^r)) = & \frac{r}{w+1}, r = 0, 1, \ldots, w+1. \end{aligned} \tag{8}$$

This solves the problem that the expected playout probability in (6) cannot reach beyond $(w/w+1)$ or below $(1/w+1)$.

Given a user-specified loss rate $\hat{\varepsilon}_l$, we are now looking for the index $\hat{r}$ and corresponding delay $D^{\hat{r}}$ that allows to achieve $\hat{\varepsilon}_l$ with the smallest possible delay. Put differently, we are looking for the greatest $D^{\hat{r}}$ such that $\mathcal{E}(F(D^{\hat{r}})) \leq 1 - \hat{\varepsilon}_l$. From (8), the corresponding index is given by $\hat{r} = \lfloor (w+1)(1 - \hat{\varepsilon}_l) \rfloor$. Given this index, the playout deadline $d_{\max}^{i+1}$ can be approximated by the interpolation between $D^{\hat{r}}$ and $D^{\hat{r}+1}$ as

$$d_{\max}^{i+1} = D^{\hat{r}} + \left(D^{\hat{r}+1} - D^{\hat{r}}\right)[(w+1)(1 - \hat{\varepsilon}_l) - \hat{r}].$$

Note that, due to the heavy-tailed nature of network delay, the maximum possible value of the delay d_n^{i+1} cannot be determined from a limited sample space. Hence, the statistic obtained from the last w samples is often too optimistic. By adding an estimate of the maximum delay, D^{w+1}, as shown by (7), the sample becomes more conservative. A higher estimate results in higher buffering delay and therefore lower loss rate.

A more accurate estimation of the delay distribution is also possible by using a larger window size w. However, this has the disadvantage that the window-based estimator will adapt less responsively to the varying network delay. Hence, the choice of w determines how fast the algorithm is in adapting to the variation and is subject to a tradeoff between accuracy and responsiveness. The values used in our experiments are given in Section VI.

One important feature of the history-based estimation is that the user can specify the acceptable loss rate, $\hat{\varepsilon}_l$, and the algorithm automatically adjusts the delay accordingly. Therefore, the tradeoff between buffering delay and late loss can be controlled explicitly. In practice, loss rates of up to 10% can be tolerated when good loss concealment techniques are employed, as discussed in more detail in Section V.

B. Adaptation to Delay Spikes

From network delay traces, it is common to observe sudden high delays ("spikes") incurred by voice packets, as packets 113–115 show in Fig. 1. Different ways of detecting such spikes and adjusting the playout time accordingly were used in [1] and [2]. However, in previous literature, due to the nature of silence-dependent time adjustment and the fact that the spike period is usually very short and unpredictable, the chance of being able to adjust the playout time when a spike is detected is very small [Fig. 1(b)]. As a result, burst packet loss resulting from spike delays cannot be effectively alleviated.

Delay spikes usually occur when new traffic enters the network and a shared link becomes congested, in which case past statistics are not useful to predict future delays. In our scheme, we therefore switch to *rapid adaptation mode* when the present delay exceeds the previous one by more than a threshold value. In rapid adaptation mode, the first packet with unpredictable high delay has to be discarded. After that, the delay estimate is set to the last "spike delay" without considering or further updating the order statistics. Rapid adaptation mode is switched off when the delays drop down to the level before the mode is in force and the algorithm returns to its normal operation reusing the state of order statistics before the spike occurred. This rapid adaptation is only possible when individual packets are scheduled and scaled as in our scheme. It is often helpful to avoid burst loss as illustrated in Fig. 1(c).

V. Low Delay Loss Concealment

Even with adaptive playout scheduling a certain number of packets will arrive after their scheduled playout time or be lost over the network. To recover the lost information as well as possible, various loss recovery techniques have been investigated in the past. A survey studying different tradeoffs among algorithm delay, voice quality and complexity is presented in [16]. We propose a new concealment method that is based on the packet scaling operations described in Section III. It is a hybrid of time-scale modification and waveform substitution, which is used to conceal both late loss and link loss by exploiting the redundancy in the audio signal. The good sound quality by time-scale modification has already been demonstrated in [10] and [11]. However, in [10] and [11], an algorithm delay of 2–3 packet times is introduced by using one-sided information and working on a block of 2–3 packets. Our method takes advantage of scaling one packet (see Section III) and using two-sided information by working together with adaptive playout scheduling. This concealment method reduces the delay to one packet time and results in better voice quality. Waveform repetition was initially proposed in [12] and [17], and is built into our scheme to repair burst loss. Waveform repetition does not introduce any algorithm delay other than a short merging period, however it does not provide as good a sound quality as time-scale modification [16].

The proposed concealment method is illustrated in Fig. 5. For the following analysis we ignore the processing time of packet scaling and assume that a packet can be scaled instantly and then played out. At the time packet i is assumed lost, its prior packet $i-1$ is extended with a target length of $2L_0$ and then played out. Because we have to wait for the receipt status of packet i, packet $i-1$ has to be buffered and delayed by one packet time. Packet i is assumed lost if it is not received by the time packet $i-1$ is to be played out, and the concealment starts at that moment. Further operation depends on the loss pattern. If packet i is the only packet lost (packets $i+1$ and $i+2$ are received by their deadlines), packet $i+1$ is extended with a target length of $1.3L_0$. Before concatenating the expanded packets $i-1$ and $i+1$, a search of similar waveform within a limited region (about half a pitch period in length) is performed to obtain the merging position. As is shown in Fig. 5(a), small segments found from either side are then weighted by falling and rising windows before merging. In this way discontinuities are avoided and better sound quality compared to a fixed merging position [10] is obtained. The total expansion of packets $i-1$ and $i+1$ will cover or, most likely, exceed the gap resulting from the lost packet. Note that uncertainty about the length of the concatenated packets remains because of the undetermined output size by WSOLA and the described merging procedure. The resulting length of modified packets is then $L^{i-1}+L^{i+1}-L_{merge}$, and the playout time of packet $i+2$ is

$$t_p^{i+2} = t_p^{i-1} + L^{i-1} + L^{i+1} - L_{merge}. \tag{9}$$

Note that a successful concealment should have $t_p^{i+2} > t_r^{i+2}$. In general, t_p^{i+2} will not match the desired playout time. Therefore, the actual playout is likely to be either ahead or behind the scheduled playout by a small difference δL, as illustrated in Fig. 5. However, this can be corrected by the adaptive playout algorithm by scaling the following packets. Therefore, the use of time-scale modification for adaptive playout as well as loss concealment allows additional flexibility and integrates nicely into the overall system.

Besides single packet loss, our concealment method can also handle interleaved loss patterns, or bursts loss, as shown in Fig. 5(b) and (c), respectively. Compared to concealment techniques discussed in [10], [11] significant improvement can be achieved for these cases. The main advantage of the proposed method is that it can detect such patterns and automatically adjust the amount of scaling. In either case, when packet i is lost, the scaling of packet $i-1$ follows the same procedure depicted in Fig. 5(a) since the future loss pattern is unknown at time i. In Fig. 5(b), once packet $i+2$ is determined to be lost, packet $i+1$ is scaled with a target length of $2L_0$ instead of $1.3L_0$, to cover the gap resulting from the second loss. In Fig. 5(c), where packets i and $i+1$ are lost, the waveform of the scaled packet $i-1$ is repeated in order to conceal burst loss. In both cases, search of similar waveforms is performed for merging, and adaptive playout time adjustment is used on the following packets if necessary.

Finally, the concealment of two or more consecutive packet losses is illustrated in Fig. 5(c). When more than two consecutive losses occur, waveform repetition is used for a maximum of three times, before the mechanism stops to generate output and resets the playout schedule. Due to the replicating nature of waveform repetition, burst loss degrades voice quality most severely, even if after being concealed. Although there exist re-

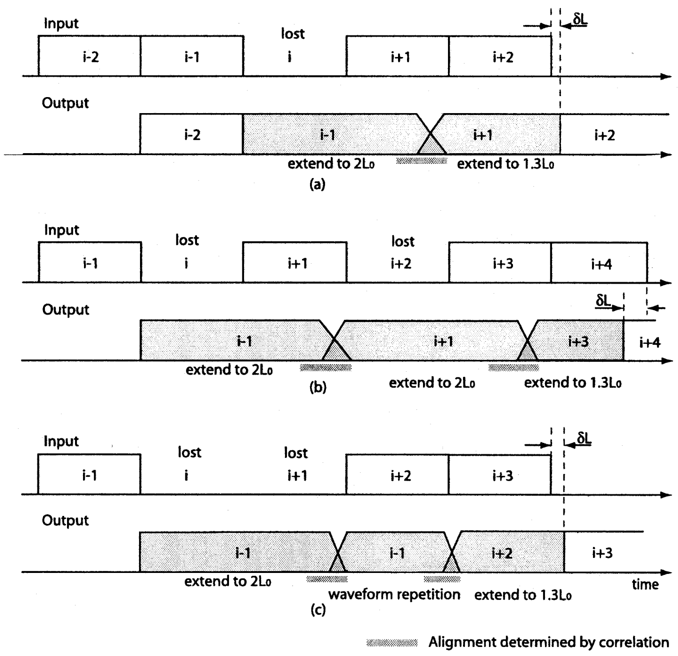


Fig. 5. (a) Loss concealment for single loss, (b) interleaved loss, and (c) consecutive loss.

construction schemes using interleaving [18] or FEC [5], [19] to protect data from burst loss, those approaches are made at the cost of higher delay. As shown in Section VI, one particular advantage of adaptive playout scheduling is its capability of reducing the loss of consecutive packets, i.e., burst loss.

The algorithm for the proposed loss concealment method is summarized in Fig. 6. This method introduces a delay of one-packet time, but offers good voice quality and handles high loss rates of various patterns. Note that when used together with adaptive playout scheduling, the playout times in Fig. 4 should be offset by one-packet. However, since other concealment methods could be used instead, the adaptive playout scheduling algorithm described in Fig. 4 is treated in a more general form without reflecting any specific loss concealment mechanism. Furthermore, if a speech codec is used for transmission that has an internal loss concealment algorithm that operates at a lower delay, it might be advantageous to use it instead of the algorithm proposed here. The option of switching off the proposed concealment mechanism is open. In general, however, we believe that the proposed scheme (including adaptive playout scheduling and loss concealment) can be well integrated with any kind of speech codec.

VI. Performance Comparison

In this section we compare the performance of the three playout scheduling schemes *Algorithm 1–3* as described in Section II and illustrated in Fig. 1. The comparison is based on packet delay traces collected from the Internet by transmitting voice streams between hosts at four different geographic locations. The four Internet links for which we collected the data are listed in Table II, and the data sequences collected from these links are referred as *Traces 1–4* respectively. The local machine is located at Stanford University, with an IP address

```
if packet $i$ received
  if packet $i-1$ received
    if packet $i-2$ received
      Output packet $i-1$ with length $L_{i-1}$;
    else
      Extend packet $i-1$ with target length $1.3L_0$;
      Output packet $i-1$;
    endif
  endif
else
  if packet $i-1$ received
    Extend packet $i-1$ with target length $2L_0$;
    Output packet $i-1$;
  else
    Repeat and output packet $i-2$ using *waveform repetition*;
  endif
endif
```

Fig. 6. Algorithm for loss concealment.

of 171.64.90.63. The measured data is one-way delay of UDP packets with payload size of 160 bytes, reflecting 20 ms G.711 coded voice packet in 8-bit quantization. Each trace runs for 180 seconds, consisting of delay data of 9000 packets. The clocks of the local and remote hosts are synchronized using the Network Time Protocol [20].

The metrics we use for the comparison between different algorithms are the late loss rate, ε_l, and the average buffering delay, d_b, as defined in Section II. These two quantities are of our major concerns since they are directly associated with the subjective quality, and they are the receiver-controllable components of the total loss rate and total delay respectively.

In our experiments, receiving and playing out of the voice packets are simulated offline using the three algorithms under comparison. Delay traces and voice packets are read in from recorded files and different playout algorithms are executed to calculate the playout time, scale voice packets if necessary, and generate output audio. In this way, different algorithms are compared under the same conditions. After the simulation of a whole trace, the loss rate and average buffering delay are calculated and plotted in Fig. 7. The continuous curves with different loss rate and buffering delay 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. The variation of the control parameter therefore illustrates the achievable tradeoff between ε_l and d_b that can be achieved with each algorithm.

The tradeoff of using different window size w is discussed in Section IV. We have tried different window sizes and chosen $w = 300$ for Algorithm 2, and $w = 35$ for Algorithm 3, in order to achieve optimal performance in terms of delay—loss tradeoffs. The use of a smaller window size for Algorithm 3 is more appropriate due to the more responsive nature of the algorithm in adapting to the network delay variation. For Algorithm 2, a bigger window size provides better performance, since the adaptation is performed less frequently and the short-term history may result in a "myopic view" of the past [2], [6]. In summary, the window size is optimized for each algorithm based on the collected trace data.

TABLE II
COLLECTED NETWORK DELAY TRACES

	Hosts Locations	Remote Host IP Address	Measurement Start Time
Trace 1	Stanford → Chicago	64.193.109.173	17:19 May 20, 00
Trace 2	Stanford → Germany	131.188.130.203	15:04 May 19, 00
Trace 3	Stanford → MIT	18.184.0.36	12:36 Aug 29, 00
Trace 4	Stanford → China	202.112.45.46	13:57 Sep 12, 00

Fig. 7 shows the total loss rate (ε) and the burst loss rate (ε_b) that can be achieved for a given average buffering delay (d_b). The link loss rate (ε_n) is shown by a horizontal dashed line, which sets the lower bound for ε. Additional loss is caused by late loss, which is under control of the playout scheduling scheme. Hence, given $\varepsilon = \varepsilon_n + \varepsilon_l$, Fig. 7 implicitly includes the late loss rate ε_l for evaluation.

In all cases, Algorithm 3 results in the lowest buffering delay for the same loss rate and hence outperforms the other two algorithms. If targeting a late loss rate of 5% for Trace 1, the average buffering delay is reduced by 40.0 ms when using Algorithm 3 instead of Algorithm 1. Comparing Algorithm 3 with Algorithm 2 the gain is still 31.5 ms. Similarly, for Traces 2, 3, and 4, the gain of Algorithm 3 over Algorithm 1 is 20.7 ms, 4.3 ms, and 28.0 ms respectively; and the gain over Algorithm 2 is 11.8 ms, 4.4 ms, and 20.0 ms respectively.

On the other hand, if allowing the same buffering delay for different algorithms, Algorithm 3 also results in the lowest loss rate. For the example of Trace 1, if the same 40 ms buffering time is consumed, the total loss rate resulting from Algorithm 3 is more than 10% lower than that from Algorithms 1 and 2. Similar reductions in loss rate are also obtained in Traces 2, 3, and 4.

More importantly, the burst loss rate also drops when using the proposed algorithm. For Trace 1, by using Algorithm 3, the burst loss rate drops from 12% to 1% at 40 ms buffering delay. As discussed above, burst loss significantly impairs voice quality even if its rate is as low as 5%. Even for Trace 3, where the gain from Algorithm 3 in terms of late loss rate and buffering delay is the smallest, the burst loss rate is 3.9% lower at 10 ms buffering delay.

The performance gain of Algorithm 3 over Algorithms 1 and 2 depends on the characteristics of the trace. Naturally, when the network delay has no variations or unpredictable variations there is no room for improvement. For example, the link between Stanford and MIT has large bandwidth, resulting in the mildest delay variations for Trace 3 and hence the lowest gain for Algorithm 3. The gain is most significant when the range of delay values is big and the correlation between packet delays allows a good estimate based on the past w packets.

VII. Subjective Listening Test Results

As seen from Section VI, the superior performance of Algorithm 3 depends on adaptive playout time adjustment, which is enabled by packet scaling. In this section we will first show the results of a subjective listening test on scaled audio. In the next part we will present subjective test results for the overall system including loss concealment and different playout scheduling algorithms.

The listening tests were carried out according to ITU-T Recommendation P.800 [21]. Each sample used for the tests consists of two short sentences, uncorrelated in meaning, and uttered

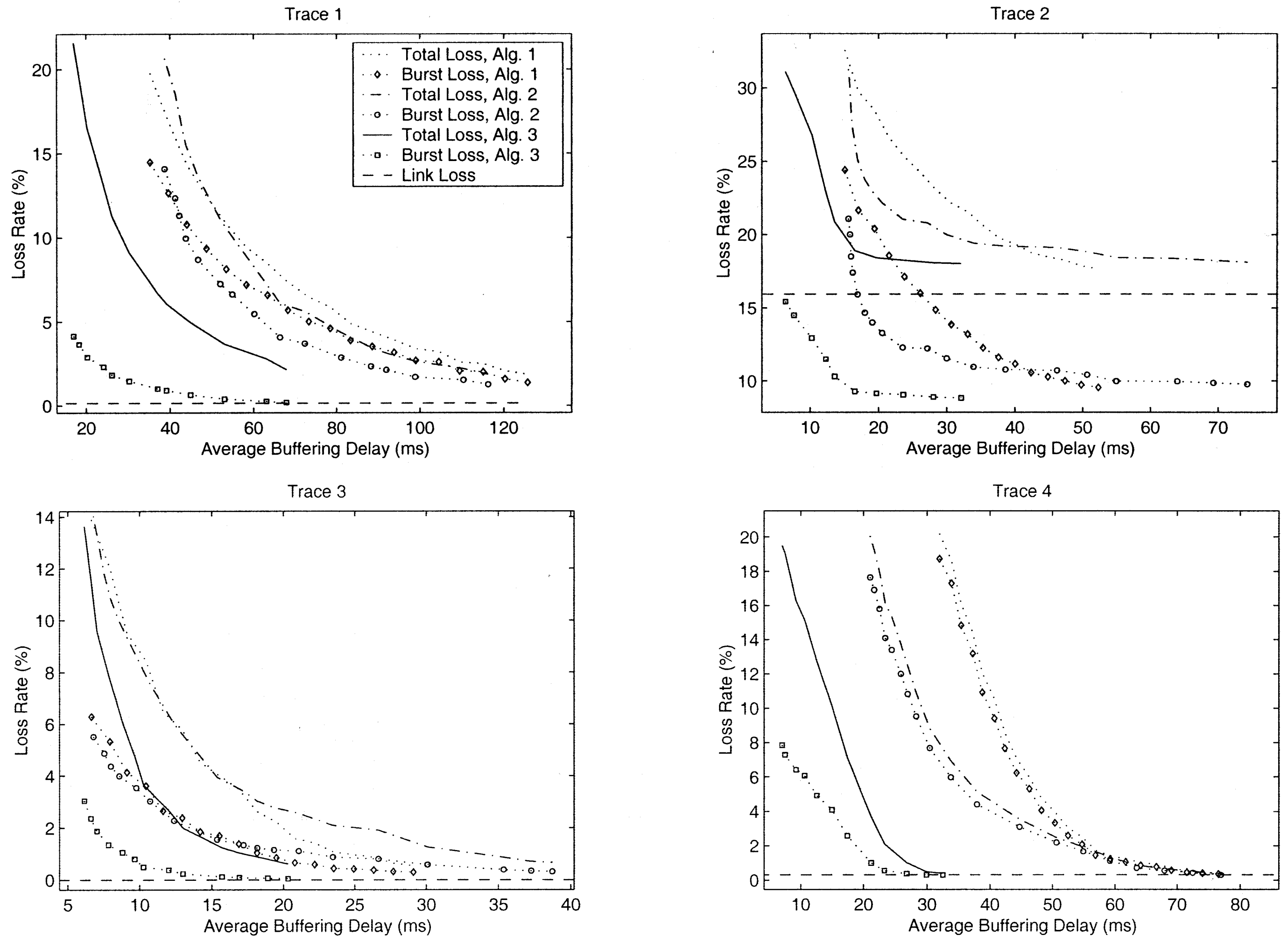


Fig. 7. Performance of playout scheduling schemes. *Algorithm 1*: fixed playout time. *Algorithm 2*: between talkspurt adjustment. *Algorithm 3*: within talkspurt adjustment.

by the same speaker. The speech samples are spoken by various male and female speakers and each sample lasts for about 6 seconds. All speech samples are sampled and digitized at 8 kHz, in 16 bit PCM format and are recalibrated to produce a speech level of -26 db relative to the peak overload level of the recording system [21]. We reemphasize that the speech samples we work on can be the PCM output of a voice decoder if any kind of codec is used. A packet contains 20 ms audio, which makes packet compression possible in most cases of all the samples simulated. Eighteen listeners participated in the test and the score for each signal processing condition is obtained by averaging the scores from all listeners and samples. In order to calibrate the obtained data, four reference conditions of modulated noise reference units (MNRU) [22] with signal-to-noise ratios of 10, 18, 24, and 30 dB are used in addition to the signal processing conditions under test.

A. Audio Scaling

To evaluate the quality degradation by scaling of voice packets, we use the *degradation category rating* (DCR) method described in Annex D of Recommendation P.800 [21]. In DCR the speech samples are presented in the format of "original sample—processed sample" to allow higher sensitivity by direct comparison of pairs. The listeners are asked to rate the degradation on a 5 to 1 scale, corresponding to *5-inaudible, 4-audible but not annoying, 3-slightly annoying, 2-annoying,* and *1-very annoying*, respectively. The scores obtained in this way are referred to as *degradation mean opinion score* (DMOS).

The processed sound samples correspond to the output of Algorithm 3 while it adapts to different network conditions. From the collected Internet traces three segments are extracted of approximately 6 seconds and with different amount of jitter. Simulating the transmission of speech samples over these network conditions using Algorithm 3, the amount and frequency of scaling is altered. The three network conditions are listed in Table III, showing that the standard deviation (STD) of network delay increases from 19.6 ms to 65.0 ms. The latter case is an extreme case of all collected data. To further characterize the delay jitter the Table III also includes the maximum jitter, which is the difference between the maximum and minimum delay in the short trace. Each test condition was tested with six samples including a null pair in which the original sample is presented twice to check the quality of anchoring. Because we want to focus on the effect of scaling, the processed samples do not carry any loss. That is, the samples are only scaled.

TABLE III
SUBJECTIVE TEST RESULTS OF PACKET SCALING

Network Condition	STD of Network Delay (ms)	Maximium Jitter (ms)	STD of Total Delay (ms)	Maximum Packet Scaling Ratio L^i/L_0	Minimum Packet Scaling Ratio L^i/L_0	Percentage of Packet Scaled (%)	DMOS
1	19.6	86.0	7.5	1.7	0.55	17.8	4.7
2	20.9	112.0	10.5	2.3	0.38	18.4	4.5
3	65.0	238.0	28.2	2.1	0.35	24.1	4.6

TABLE IV
SUBJECTIVE TEST RESULTS OF PLAYOUT ALGORITHMS 2 AND 3

Trace	STD of Network Delay (ms)	Maximium Jitter (ms)	Link Loss Rate (%)	Algorithm	STD of Total Delay (ms)	Buffering Delay (ms)	Total Loss Rate (%)	Burst Loss Rate (%)	MOS
1	23.7	130.0	0	Algorithm 2	0	55.1	8.5	6.0	2.6
				Algorithm 3	7.6	54.6	2.8	0.7	3.7
2	15.9	86.0	8.3	Algorithm 2	0	26.6	13.3	4.1	2.4
				Algorithm 3	8.5	26.1	8.3	0	2.8
3	5.9	39.0	0	Algorithm 2	0	23.1	2.6	0.6	3.3
				Algorithm 3	2.6	23.0	0.3	0	4.3
4	13.7	47.0	0	Algorithm 2	0	28.8	5.1	3.9	3.0
				Algorithm 3	7.4	25.7	1.1	0	4.1

As can be seen from Table III, the degradation due to audio scaling is between *inaudible* and *not annoying*, even for extreme cases. The null pair condition received a score of 4.8, which indicates the validity of the testing methodology. In summary, these results substantiate the good quality of scaled audio, which may seem surprising considering the minimum and maximum scaling ratios of 35–230% listed in columns 5 and 6. One reason for this finding is that packets actually do not have to be scaled very frequently as shown in column 7. Even for Condition 3, fewer than 25% of the packets were actually scaled to satisfy the timing requirement of adaptive scheduling.

Also listed in Table III is the STD of the total delay within talk-spurts, which characterizes the variation of the playout rate, or playout jitter. For Algorithm 1, this 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 a comparison of columns 2 and 4.

B. Overall Audio Quality

In this section we present subjective test results that investigate the overall quality of speech using adaptive playout in combination with loss concealment. The presented samples are generated using Algorithm 2 and 3, which are the more advanced algorithms with higher practical importance. For both cases the loss concealment mechanism described in Section V is used to repair packet loss. Hence, we focus on the performance gain by adaptive playout. Similar to the Section VII-A we use four short segments from Trace 1–4 that last for approximately 6 s. The corresponding network characteristics for these segments are summarized in column 2–4 of Table IV.

Absolute category rating (ACR) is used to rate the processed samples, according to the procedure in Annex B of Recommendation P.800 [21]. In contrast to DCR, no original sample is provided for direct comparison and the listeners are asked to rate the quality of speech using an absolute scale of 5 to 1 corresponding to *5-excellent, 4-good, 3-fair, 2-poor,* and *1-bad* quality, respectively. The mean opinion scores (MOS) obtained by averaging the scores from all listeners and four different samples are listed in Table IV.

Before discussing the MOS scores, we note that Algorithm 2 and 3 have to operate at the same average buffering delay in order to allow a fair comparison. As can be seen from column 7 in Table IV, this requirement is roughly met for each trace by adjusting the control parameter appropriately. In order to obtain reasonable sound quality for each trace, the used buffering delay is adjusted to the jitter characteristics. Therefore, the highest buffering delay is used for Trace 1 while the buffering delay for Trace 3 is close to the minimum buffering delay of one packet time (20 ms), which is required for loss concealment.

For Trace 1, 2, and 4, the gain by using Algorithm 3 is more than one MOS, which indicates significant improvement of the speech quality. For Trace 2, since the link loss is dominant and due to the use of the same loss concealment technique, the difference is smaller. The MOS scores are well correlated with the loss rate and burst loss rate. For example, the low scores for both algorithms when using Trace 2 can be explained by the high total loss rate. On the other hand, the gain of Algorithm 3 compared to Algorithm 2 is a direct result of the reduced loss rate (for the same buffering delay). Note that burst loss, even at a low rate of 5%, impairs speech quality significantly. Nevertheless, Algorithm 3 is able to greatly reduce burst loss in all cases, which is directly reflected in the MOS scores.

For completeness we also include the test results of the MNRU reference conditions and unprocessed, original speech in Table V. These numbers are provided to allow an anchoring of our test group compared to other groups, which is useful for the reproduction and comparison with other work.

VIII. CONCLUSIONS

In this paper the use of WSOLA based time-scale modification techniques is investigated for adaptive playout scheduling

TABLE V
SUBJECTIVE TEST RESULTS OF MODULATED NOISE REFERENCE UNITS (MNRU) AND ORIGINAL SPEECH

SNR (dB)	10	18	24	30	Original Speech
MOS	1.4	2.7	3.7	4.1	4.4

and loss concealment in the context of real-time voice communication over IP networks. Since the proposed methods are receiver-based, no cooperation of the sender or the network is required, which makes our work widely applicable.

The adaptive playout scheduling scheme estimates the network delay based on short-term order statistics covering a relatively small window, e.g., the past 35 packets. In the case of delay spikes, a special mode is used to follow delay variations more rapidly. Given the estimate, the playout time of the voice packets is adaptively adjusted to the varying network statistics. In contrast to previous work, the adjustment is not only performed between talkspurts, but also within talkspurts in a highly dynamic way. Proper reconstruction of continuous playout speech is achieved by scaling individual voice packets using a *Single Packet WSOLA* algorithm that works on individual packets without introducing additional delay or discontinuities at packet boundaries. Results of subjective listening tests show that the DMOS score for this operation is between *inaudible* and *audible but not annoying*. This negligible quality degradation can also be observed for extreme network conditions that require scaling ratios of 35–230% for up to 25% of the packets.

Simulation results based on Internet measurements show that the tradeoff between buffering delay and late loss can be improved significantly. For a typical buffering delay of 40 ms, the late loss rate can be reduced by more than 10%. More importantly, the proposed algorithm is very well suited to avoid the loss of multiple consecutive packets, which is particularly important for loss concealment. For example, the burst loss rate can be reduced from 12 to 1% at 40 ms buffering delay, which results in significantly improved audio quality.

A WSOLA based loss concealment technique is also proposed to combat loss, and work together with adaptive playout scheduling. Compared to previous work, the proposed scheme operates at very low delay, i.e., one packet time, and can handle various loss patterns more effectively. It is shown that loss concealment can take advantage of the flexibility provided by adaptive playout, and hence, the use of time-scale modification for adaptive playout scheduling as well as loss concealment integrates seamlessly into the overall system.

Finally, the overall audio quality is investigated based on subjective listening test comparing Algorithm 2 (between talkspurt adjustment) with the proposed Algorithm 3 (within talkspurt adjustment) under the same buffering delay and loss concealment technique. Though the achievable gain depends on the network trace, typical gains are one MOS point on a 5-point scale, e.g., from *fair* to *good* audio quality. Considering that this significant improvement is achieved at the receiver side only, we expect adaptive playout scheduling to become widely adopted in the near future.

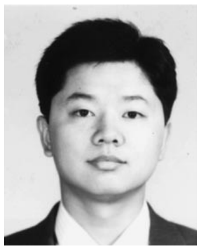
ACKNOWLEDGMENT

The authors would like to thank A. Stenger for providing us the source code of his earlier work on time-scale modification, B. Prabhakar and F. A. Tobagi for helpful discussions on network related issues, and Z. P. Tang for developing the tools for network delay measurements and helping the experiments. They also thank B. Westrich, X. Yang, and N. Yang for allowing us to use their machines to collect the delay traces, and all the listeners for participating in our subjective tests.

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*, vol. 2, June 1994, pp. 680–688.
[2] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: Performance bounds and algorithms," *Multimedia Syst.*, 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 Conf. Local Computer Networks*, Oct. 1999, pp. 224–231.
[4] P. DeLeon and C. J. Sreenan, "An adaptive predictor for media playout buffering," in *Proc. ICASSP'99, IEEE Int. Conf. Acoustics, Speech, and Signal Processing.* , vol. 6, Mar. 1999, pp. 3097–3100.
[5] J. Rosenberg, L. Qiu, and H. Schulzrinne, "Integrating packet FEC into adaptive voice playout buffer algorithms on the internet," in *Proc. IEEE INFOCOM 2000*, vol. 3, Tel Aviv, Israel, Mar. 2000, pp. 1705–1714.
[6] Y. Xie, C. Liu, M. Lee, and T. N. Saadawi, "Adaptive multimedia synchronization in a teleconference system," *Multimedia Syst.*, vol. 7, no. 4, pp. 326–337, July 1999.
[7] C. J. Sreenan, J.-C. Chen, P. Agrawal, and B. Narendran, "Delay reduction techniques for playout buffering," *IEEE Trans. Multimedia*, vol. 2, pp. 88–100, June 2000.
[8] Y. J. Liang, N. Färber, and B. Girod, "Adaptive playout scheduling using time-scale modification in packet voice communications," in *Proc. ICASSP'01* , vol. 3, Salt Lake City, UT, May 2001, pp. 1445 –1448.
[9] 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*, vol. II, Apr. 1993, pp. 554–557.
[10] A. Stenger, K. Ben Younes, R. Reng, and B. Girod, "A new error concealment technique for audio transmission with packet loss," in *Proc. Eur. Signal Processing Conf.*, vol. 3, Sept. 1996, pp. 1965–8.
[11] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod, "A new technique for audio packet loss concealment," in *Proc. IEEE GLOBECOM*, Nov. 1996, pp. 48–52.
[12] D. J. Goodman, G. B. Lockhart, O. J. Wasem, and W.-C. Wong, "Waveform substitution techniques for recovering missing speech segments in packet voice communications," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 1440–1448, Dec. 1986.
[13] W.-T. Liao, J.-C. Chen, and M.-S. Chen, "Adaptive recovery techniques for real-time audio streams," in *Proc. IEEE INFOCOM 2001*, vol. 2, Anchorage, AK, Apr. 2001, pp. 815–823.
[14] J. F. Gibbon and T. D. C. Little, "The use of network delay estimation for multimedia data retrieval," *IEEE J. Select. Areas Commun.*, vol. 14, no. 7, pp. 1376–1387, Sept. 1996.
[15] R. V. Hogg and E. A. Tanis, *Probability and Statistical Inference*, 5th ed. New York: Macmillan, 1997.
[16] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, pp. 40–48, Sept.–Oct. 1998.
[17] O. J. Wasem, D. J. Goodman, C. A. Dvorak, and H. G. Page, "The effect of waveform substitution on the quality of PCM packet communications," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 342–348, Mar. 1988.
[18] B. W. Wah and D. Lin, "Transformation-based reconstruction for real-time voice transmissions over the internet," *IEEE Trans. Multimedia*, vol. 1, pp. 342–351, Dec. 1999.
[19] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for internet telephony," in *Proce. IEEE INFOCOM '99*, vol. 3, Mar. 1999, pp. 1453–1460.

[20] D. Mills, "Internet Time Synchronization: The Network Time Protocol," Internet Engineering Task Force, RFC-1129 , 1989.
[21] "Methods for Subjective Determination of Transmission Quality,", ITU-T Rec. P.800, 1996.
[22] "Modulated Noise Reference Unit (MNRU),", ITU-T Rec. P.810, 1996.



Yi J. Liang (S'97–M'02) received the B.Eng. degree from Tsinghua University, Beijing, China, and the Ph.D. in electrical engineering from Stanford University, Stanford, CA, in 2003.

From 2000 to 2001, he conducted research with Netergy Networks, Inc., Santa Clara, CA, on voice over IP systems that provide improved quality over best-effort networks. Since 2001, he has been the lead of the Stanford-Hewlett Packard Labs' low-latency video streaming project, in which he developed error-resilience techniques for rich media communication at low latency over IP networks. In the summer of 2002 at HP Labs, Palo Alto, CA, he developed an accurate loss-distortion model for compressed video and contributed in the development of the mobile streaming media content delivery network (MSM-CDN) that delivers rich media over 3G wireless networks. His expertise is in the areas of networked multimedia systems, real-time voice communication, and low-latency multimedia streaming over the wireline and wireless networks.

Nikolaus Färber (M'02) received a Dipl.-Ing. degree (Dipl.-Ing.) in electrical engineering from the Technical University of Münich, Germany, in 1993. In October 1993, he joined the Telecommunications Laboratory, University of Erlangen-Nuremberg, Germany, and received the Dr.-Ing. degree in March 2000. His doctoral thesis is entitled "Feedback-Based Error Control for Robust Video Transmission."

He was with the research laboratory Mannesmann Pilotentwicklung in 1993, where he developed system components for satellite based vehicular navigation. After visiting Stanford University as a Post-Doctoral student, he joined the Speech Coding group of Ericsson Research, Nuremberg, Germany, in June 2001. His research on robust video transmission started as a Visiting Scientist at the Image Processing Laboratory, University of California, Los Angeles. Since then, he has published several conference and journal papers on the subject and has contributed successfully to the ITU-T Study Group 16 efforts for robust extensions of the H.263 standard. He performed research for 8×8, Inc., Santa Clara, CA, and RealNetworks, Inc., Seattle, WA. At Stanford University, he was leading a project on Voice over IP in cooperation with Netergy Networks, Santa Clara. In this project, adaptive playout and TCP window control was investigated for VoIP in legacy LAN environments. His work at Ericsson includes the standardization of Tandem Free Operation (TFO) in 3GPP and off-line coding of speech for streaming and multimedia messaging.

Dr. Färber served as the Publicity Vice Chair for ICASSP-97 in Münich.

Bernd Girod (M'80–SM'97–F'98) received the M.S. degree in electrical engineering from the Georgia Institute of Technology (Georgia Tech), Atlanta, in 1980 and the Doctoral degree (with highest honors) from University of Hannover, Hannover, Germany, in 1987.

Until 1987, he was Member of the Research Staff, Institut für Theoretische Nachrichtentechnik und Informationsverarbeitung, University of Hannover, working on moving-image coding, human visual perception, and information theory. In 1988, he joined Massachusetts Institute of Technology, Cambridge, first as a Visiting Scientist with the Research Laboratory of Electronics, then as an Assistant Professor of Media Technology at the Media Laboratory. From 1990 to 1993, he was Professor of Computer Graphics and Technical Director of the Academy of Media Arts, Cologne, Germany, jointly appointed with the Computer Science Section of Cologne University. He was a Visiting Adjunct Professor with the Digital Signal Processing Group at Georgia Tech in 1993. From 1993 until 1999, he was Chaired Professor of Electrical Engineering/Telecommunications at the University of Erlangen-Nuremberg, Germany, and the Head of the Telecommunications Institute I, co-directing the Telecommunications Laboratory. He served as the Chairman of the Electrical Engineering Department from 1995 to 1997, and as Director of the Center of Excellence "3-D Image Analysis and Synthesis" from 1995–1999. He was a Visiting Professor with the Information Systems Laboratory of Stanford University, Stanford, CA, during the 1997/1998 academic year. Currently, he is a Professor of Electrical Engineering in the Information Systems Laboratory, Stanford University. He also holds a courtesy appointment with the Stanford Department of Computer Science. His research interests include networked multimedia systems, video signal compression, and 3-D image analysis and synthesis. As an entrepreneur, he has worked successfully with several start-up ventures as founder, investor, director, or advisor. Most notably, he has been a founder and Chief Scientist of Vivo Software, Inc., Waltham, MA (1993–1998); after Vivo's acquisition, since 1998, Chief Scientist of RealNetworks, Inc. (Nasdaq: RNWK); and, since 1996, an outside Director of 8×8, Inc., Santa Clara, CA (Nasdaq: EGHT). He has authored or co-authored one major textbook and over 200 book chapters, journal articles and conference papers in his field, and he holds about 20 international patents.

Dr. Girod has served as on the Editorial Boards or as Associate Editor for several journals in his field, and is currently Area Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, as well as member of the Editorial Boards of EURASIP's *Signal Processing*, the *IEEE Signal Processing Magazine*, and the ACM *Mobile Computing and Communication Review*. He has chaired the 1990 SPIE conference on "Sensing and Reconstruction of Three-Dimensional Objects and Scenes" in Santa Clara, CA, and the German Multimedia Conferences in Münich, Germany, in 1993 and 1994, and has served as Tutorial Chair of ICASSP-97 in Münich and as General Chair of the 1998 IEEE Image and Multidimensional Signal Processing Workshop in Alpbach, Austria. He has been the Tutorial Chair of ICIP-2000, Vancouver, BC, Canada, and the General Chair of the Visual Communication and Image Processing Conference (VCIP), San Jose, CA, in 2001. He has been a member of the IEEE Image and Multidimensional Signal Processing Committee from 1989 to 1997 and was elected Fellow of the IEEE in 1998 "for his contributions to the theory and practice of video communications." He was named Distinguished Lecturer for the year 2002 by the IEEE Signal Processing Society.