Adaptive Playout Scheduling with Audio Time-Scaling for the Nomadik Multimedia Processor

Antonio Servetti(1), Juan Carlos De Martin(2), Giulio Urlini(3), Andrea Lorenzo Vitali(3)

(1) Politecnico di Torino,
(2) IEIIT-CNR[1],
(3) STMicroelectronics

Some of the major challenges for interactive and streaming multimedia applications are the delay and delay jitter of today's IP packet networks.
In this paper, we present the implementation, on the STMicroelectronics Nomadik™ multimedia processor family, of an advanced playout scheduling scheme that achieves high robustness against late packets while maintaining very low buffering delay at the receiver.
We modified the porting on Nomadik of a common VoIP application in order to enable the adaptation of the playout time of each speech frame to the estimated network delay. Proper handling of time compression and expansion of speech frames is achieved using a time-scale modification technique that does not alter the frame frequency content. The same technique has also been adopted for the concealment of packet losses.

1. Introduction

In traditional wireline Internet or in the emerging wireless data services, smooth and timely delivery of audio is an essential requirement for Quality of Service (QoS) provision. There are many factors that could affect QoS of a packet audio transmission. Among them, end-to-end delay, delay jitter, and packet loss are some of the major issues. Usually, at the receiver, a *playout buffer* with limited capacity is used to smooth out delay variation of received packets. Playout, in fact, does not start just after the reception of the first packet, but a pre-roll delay is introduced to fill the buffer to the desired level. However, packets that arrive too early and cannot be accommodated in the buffer if it is full or packets that arrive too late and miss their playout deadline are still discarded at the receiver, thus increasing the overall loss rate of the application.

Research on packet-based audio and video communications over best-effort networks has intensively investigated methods to support both low latency and reduction of late losses. Note that there is a tradeoff between the average time that packets spend in the playout 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

1. Prof. Juan Carlos De Martin is now with the Politecnico di Torino, Turin, Italy

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. Two way communications are very sensitive to the system latency, so end-to-end delay has to be kept within a very low range to allow seamless interactivity. In streaming, larger delays have been traditionally considered acceptable - especially in the case of PC-based streaming - but, recently, a new class of streaming applications is emerging; the most common scenario is probably set-top-box-based Video-on-Demand over broadband networks. In that case, customers expect a TV-like experience, that is, a very short delay when flipping channels. To guarantee that kind of reactivity, a streaming infrastructure also needs to operate with an overall low delay and should reduce the pre-roll delay.

One of the techniques available to reduce overall delay is to employ adaptive-size media playout buffers. Instead of determining the size of the playout buffer proportionally to the expected average standard deviation of the delay and then keeping it constant, the size can follow the instantaneous standard deviation of the delay, minimizing the average delay introduced by the buffer. One problem caused by the introduction of a variable delay element right before playout is that the client needs to vary the rate at which it plays out the audio-visual stream. To do so, a technique to time-scale the duration of audio frames must be available at the receiver. With this technique, the pre-roll delay may also be reduced by playing the media more slowly than normal just after the reception of the first packet and then allowing the buffer to fill to the target level over time.

In this paper, we present an implementation of state-of-the-art adaptive playout and time-scaling techniques for speech signals on the *STMicroelectronics' Nomadik* multimedia application processor, with particular attention to the issues related to the scenario of multimedia transmission over wired and wireless IP networks. Several algorithms are known in the literature for audio scaling in the time and frequency domain; our application achieves good quality results with the computationally efficient overlap-add technique based on waveform similarity. Besides its use to enable adaptive playout buffering, the time-scaling technique has also been used as packet loss concealment scheme. Missing frames are generally concealed with the replication of the last received frame, but better results may be achieved by extending neighboring frames to fill the gap corresponding to lost frames.

This document is organized as follows. Section 2 describes the basic idea of adaptive playout. Section 3 describes how audio frames can be scaled using time-scaling techniques. In Section 4, a loss concealment mechanism that works together with the adaptive playout scheme is proposed. Then, in Section 5, we present the implementation of the adaptive playout scheduling and loss concealment technique on the STMicroelectronics Nomadik multimedia processor. Conclusions are drawn in Section 6.

2. ADAPTIVE SCHEDULING OF AUDIO PLAYOUT

Commonly, in order to absorb short-term fluctuations in the data arrival rate, multimedia packets are not played out immediately after reception, but they are held in the so-called playout buffer until their scheduled playout deadline arrives. This buffer is also known as *dejitter buffer* since the buffering absorbs variations in end-to-end delay (delay jitter) that may cause playout interruptions. The longer the buffer size, the greater the protection against such interruptions. However, the buffering increase has a cost in terms of transmission latency that progressively reduces the interactivity level of the communication.

Research on Internet multimedia transmission has largely focused on error concealment and delay control in the presence of delay jitter and packet loss [1][2]. Given a fixed receiver buffer and a tight end-to-end delay bound, some packets sent to the receiver may still be discarded if the receiver buffer is not adjusted to accommodate the time varying end-to-end delay. For example, late arrival packets, which arrive after their scheduled playout time, are discarded (*late loss*). On the contrary, it may sometimes happen that a packet that arrives too early, for its scheduled playout has to be discarded because there is no place to hold it in the buffer (*early loss*). Delay variations happen when packets experience different queuing delays in network apparatuses because of the varying Internet load. Packet droppings at the receiver caused by network delay jitter increase the overall packet loss rate experienced by the user, and thus result in increased degradation of audio playout.

While common algorithms for managing a playout buffer adopt the fixed approach (i.e., they assume that the range of network delays is predictable and use a static buffer size and schedule), a more effective solution is to adopt the reactive approach, that is, to measure the delay jitter and use that information to adjust the buffer size dynamically and schedule to avoid lateness. The latter approach records the reception time of each packet and uses this information to make short-term predictions about network delay with the aim of implementing an application-controlled tradeoff of packet lateness against buffering delay suitable for applications which demand low delay but can tolerate or conceal a small amount of late packets. In applications like packet telephony, for example, large delays cause a significant negative impact on quality because they reduce interactivity, i.e., delays in the range of 150-250 milliseconds are often quoted as being the maximum acceptable total end-to-end delay [3]. In this case, a small percentage of losses may be given up in return for a lower end-to-end delay and an increased interactivity.

Among the reactive approaches, it has been proposed, for voice telephony, to adjust the playout buffer size adaptively only during silence periods [4][5]. This technique is based on the observation that, for a typical conversation, the conversation can be grouped into talkspurts separated by silence periods and that active speech periods represent only approximately 40% of a conversation. Thus, the playout time of a new talkspurt may be adjusted by extending or compressing the amount of silence without any modification of the voice packets. With this technique, most late-arriving packets can be played out instead of being thrown away. However, adaptive playout applied solely on silence intervals could only work if the network is relatively stable (which means that the statistical estimation based on the previous talkspurt could hold for the current) and the employed silence detection algorithm is effective. This is not the case when talkspurts are long and network delay variation is high within talkspurts, that is, if a delay spike happens in the middle of a talkspurt, the algorithm has to wait until the next spurt to adjust the delay. Thus, the performance result heavily depends on speech contents and on the network situation[2].

In this work, we employ a novel adaptive playout algorithm that exploits a time-scale audio modification technique to extend the previous algorithm based on silence intervals [6][7]. With the ability of extending or compressing the duration of each individual frame, the playout schedule may be adaptively adjusted (according to the varying network conditions) in a more dynamic and reactive way, i.e., not only during silence periods but also during talkspurts. The continuous output of high quality audio is achieved by the scaling of the audio frames using a time-scale modification technique. By applying a varying degree of stretching for each frame, every frame could contribute in adapting to the network delay jitter as well as to the packet loss. In this work, for time-scale modification, a packet-based version of the waveform similarity overlap-and-add (WSOLA) scheme is adopted [8]. The time-scale modification factor is estimated for each frame depending on the delay constraint, delay statistics, and the number of late-arrival packets[3].

Improvement of the tradeoff between buffering delay and late loss is based on the interesting finding that the variation in the playout time-scale (i.e., slowing down and speeding up the playout) introduced by the audio time-stretching technique is

2. In this context, we do not consider any particular implementation of silence detection algorithms. We just assume that the decoder is provided with a method to discriminate between active and non-active speech segments.

3. In this document, the words packet and frame are sometimes used interchangeably because it is common, in many VoIP applications, to encapsulate only one frame per RTP packet. For simplicity, when this is not the case (i.e., an RTP packet contains several frames), we assume to extend the meaning of the word “frame” to the overall audio content of the packet. The word “packet” will be always preferred when considering network related issues such as end-to-end delay, arrival time, etc.

perceptually tolerable if the audio is appropriately processed, as demonstrated by subjective listening tests [9].

3. AUDIO TIME-SCALE MODIFICATION

Scaling of audio frames, as previously introduced, is necessary to achieve high quality continuous output in audio transmission when each individual frame playout schedule is adaptively adjusted according to the varying network conditions, even during talkspurts.

Appropriate processing of the audio signal is necessary to ensure perceptually acceptable results. The problem of time-scaling an audio signal lies in the corresponding frequency distortion. It is common experience that when an audio signal is played back at a higher speed than the recording speed, the resulting sound is distorted in that its pitch is raised. We, instead, require a scaling of the perceived timing attributes, such as speaking rate, without affecting the perceived frequency attributes, such as pitch.

Because of the mathematical duality between time domain and frequency domain representations of audio, and from the properties of the Fourier transform [10], we can consider two equivalent formulations of this problem:

- modify the time domain representation of the audio signal without altering its perceived frequency attributes; and
- modify the frequency domain representation of the audio signal without altering its perceived time structure.

In this work, due to the computational constraints imposed by the embedding of this technique on the Nomadik multimedia processor, we consider the problem in its first formulation. In fact, time-scaling methods that use waveform-similarity and overlap-add (OLA) construction methods may achieve an attractive quality at a low computational cost. The frequency approach is more computational intensive and less suited to low processing delay requirements of packet based real-time communications [11].

With OLA synthesis, time-scale modification can be realized using time domain operations only. The modification is obtained by excising segments from the input signal and repositioning them along the time axis before reconstructing the output signal by weighted overlap-addition of the segments. The repositioning of segments corresponds to the introduction of a linear phase difference between the individual segments that disrupts the periodical structure of audio, unless proper synchronization is ensured by choosing an offset equal to a multiple of the waveform “virtual” period. As a consequence, each input segment should be realigned to the already formed portion of the output signal before performing the OLA operation [12].

In 1993, Verhelst proposed the waveform-similarity and overlap-add (WSOLA) algorithm [13], a synchronization strategy inspired on a waveform similarity criterion. In this technique, proper synchronization is ensured during segmentation instead of during superposition. The audio input signal is first decomposed into overlapping segments of equal length and these segments are then realigned and superimposed with equal and fixed overlap to form the output signal. 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. This technique was then modified and improved in [14] for packet loss concealment and in [6] for working on only *one* frame so that each frame of a real-time stream could be scaled immediately without introducing any additional processing delay.

An illustration of the WSOLA blockwise operation technique for slowing-down audio is presented in Fig. 1. The graph at the top corresponds to the input audio, and the one at the bottom corresponds to the time-scaled output audio. In the modified WSOLA algorithm, the output signal is constructed by taking small sections of the input audio frame from different time points (x_k) and “pasting” them together to form the output audio frame at time points (y_k). To achieve smooth transitions, each segment

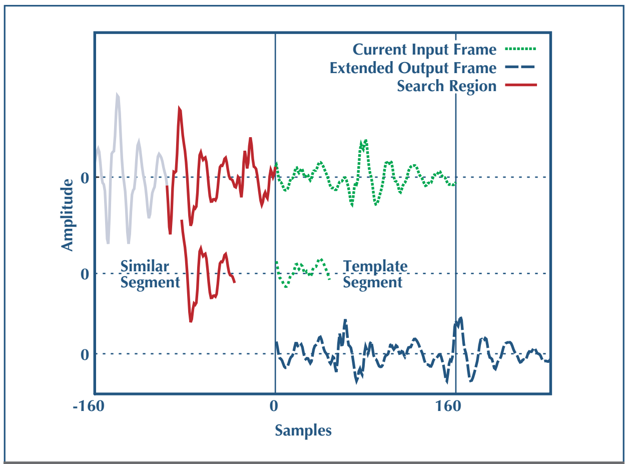

Figure 1: Frame time-scale modification example. The current frame (top green waveform) is extended to produce a modified output frame (bottom blue waveform). A template segment is used to find a similar segment in the previous output (top red waveform). The current frame is then overlap-added with the scaled waveform (we can see the effect of the weighting function).

is weighted with a Hanning window *w[n]* such that the overlap-add with the already constructed output waveform will preserve the signal continuity. This procedure is given by the following synthesis equation

$$q[n] = \sum_k w[n - y_k] \cdot v[n - y_k + x_k], \qquad (1)$$

where *k* is the step index, *v[n]* is the input signal, and *q[n]* is the output signal.

For example, assume that the green waveform in Fig. 1 represents the current frame (e.g., 20 ms speech waveform), while the black and red waveform is the previous frame that has already been processed and played out. Our objective is to extend the time duration of the current speech segment and thus its playout time. To scale the speech frame, we first select a *template segment* of constant length in the input and then search backwards for the *similar segment* that exhibits maximum similarity to the template segment. Since WSOLA has been limited to operate on a single frame, it cannot change the previous or next frame. Thus, the original algorithm has been modified to use multiple successive template segments of shorter length (inside the current frame) and to position the first template segment at the beginning of the current frame (and not before), as shown in the figure. Since we are slowing down the playout time, the similar segment must be selected from a time point inside a *search region* (red waveform) in the past. The beginning of the search region is calculated in advance and is a function of the required time-scaling factor. We cannot modify a previous frame that might already be played out, but a similar waveform may still be extracted from it to construct the new output. In our case, the cross correlation

$$c_i = \sum_{j=0}^{\Delta y - 1} v[i + j] \cdot v[x_{k-1} + \Delta y + j], \qquad (2)$$

a common distance measure in audio signal processing, is used to select the best candidate segment inside the search region (Δy) that is closer to the template. Once the similar segment is found, it is copied, with all the following samples, to the new location and overlap-added with the template segment to generate the output. The resulting output (bottom blue waveform) is longer than the input due to the shift of the similar segment relative to the template segment. We can observe from the output waveform that one extra pitch period is created and added. However, it is not just a simple replication of any pitch period from the input but the mixing of several pitch periods instead. This explains why the sound quality using time-scale modification is good. For voice communications, subjective listening tests [9] show, in fact, that infrequent scaling does not degrade the speech quality, even if the scaling ratio is occasionally high. Moreover, this scheme is entirely independent from the audio or speech codec used because the operations are applied on the PCM output of the decoder. However, it is important to note that this process heavily relies upon the existence of a quasi-periodic waveform, as in speech and monophonic music. Thus, it is generally sub-optimal for most polyphonic music, due to the complex multi-pitch nature of those waveforms. Nevertheless, enhancements of this approach such as sub-band WSOLA [15], which apply this time-domain time-scale modification algorithm on a sub-band basis, have been shown to provide adequate sound quality for any kind of audio, including news magazines, cinema movies, and music TV.

4. PACKET LOSS CONCEALMENT WITH AUDIO TIME-SCALING

Audio adaptive playout techniques aim at reducing the rate of late losses; however, it is still possible that a certain number of packets will arrive after their scheduled playout time. Additionally, audio packets may be lost along the communication path due to network congestion. Sound quality at the receiver can then be improved by error concealment, which is transmitter independent. There exist a number of proposals for concealment of lost audio frames that fundamentally try to substitute the missing signal segment by *repeating a prior segment*. These techniques generally lead to echoing or tinny sounds. Our concealment algorithm is instead based on the frame-scaling operations described in Section 3. It is a hybrid of time-scale modification and waveform substitution, the latter of which is used to conceal both late loss and link loss by exploiting the redundancy in the audio signal [14]. To avoid common distortions, the time-scale of a preceding signal segment is modified such that missing frames are covered by the *extended version of the preceding waveform*. An important demand of audio communication is low delay, which can only be achieved by modifying a small number of packets during the time-scaling. Therefore, we take advantage of the single frame time-scaling algorithm [6] to reduce the delay to one frame time, and we employ two-sided information to achieve good reconstruction quality.

Fig. 2 shows the principle of error concealment using time-scale modification of correctly received frames. At the time the current frame (red waveform) is assumed lost, the last received frame (green waveform) is extended with a scaling factor of two. Then it can be played out for an overall time of approximately two frame periods so that the missing gap caused by the lost frame is covered (blue waveform). It is important to note that, because we can scale the previous frame only after we know if the current frame has been received or not, then the previous frame has to be buffered and delayed by one frame time. Further operations depend on the loss pattern:

- if only the current frame is lost, the next frame will be extended with a scaling factor of 1.3 (because of a successive the merging operation of the two waveforms); and
- if the current and next frame are both lost, the waveform of the previous scaled frame is repeated to conceal the burst loss.

In both cases, distortions due to discontinuities at the boundary of the substituted and the next received frame are reduced by overlapping them in the range of a few milliseconds using Hanning windows. A search of similar waveform within a limited region is performed to obtain the merging position. The total expansion of the frame will not exactly cover the gap resulting

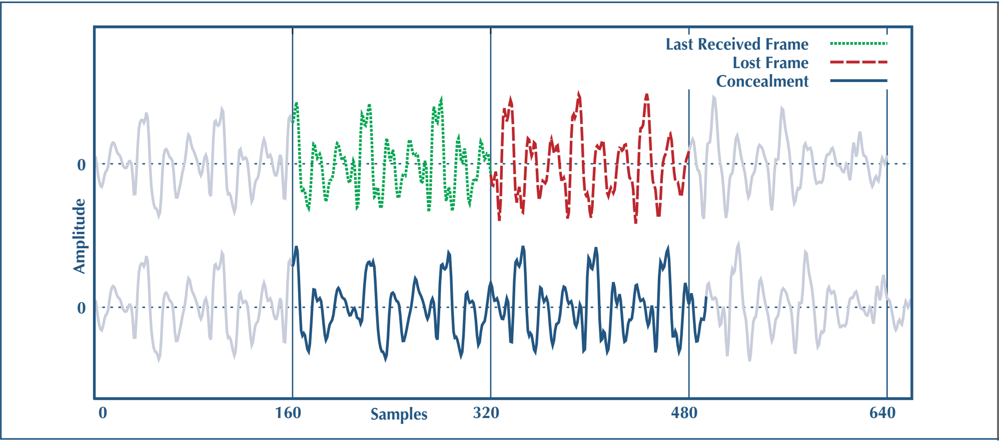

Figure 2: Example of single frame loss concealment. The output waveform is plotted below the input waveform. A frame (red) is lost, and the gap is covered by the extension (blue) of the previous frame (green).

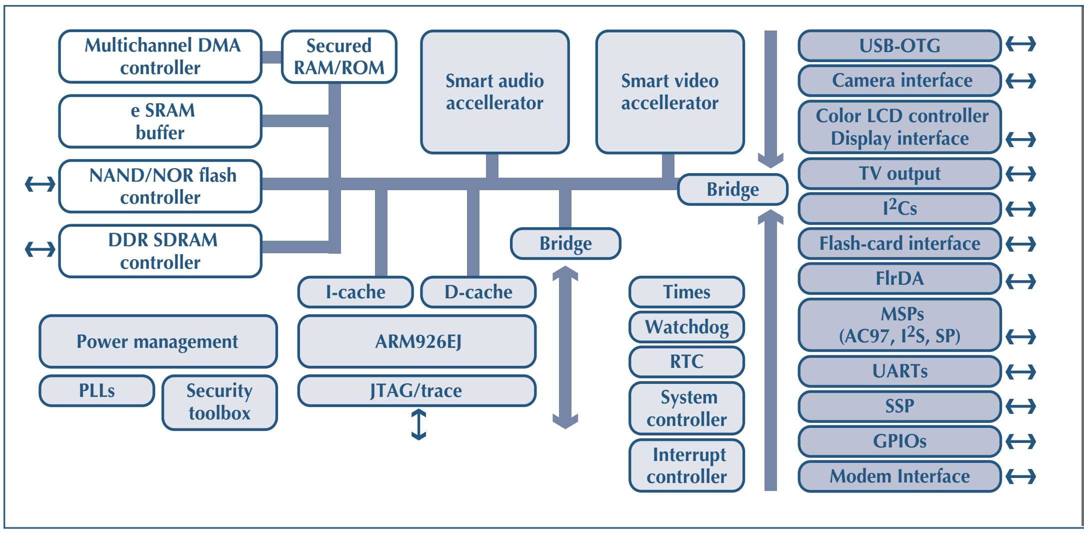

Figure 3: STMicroelectronics Normadik© architecture [16].

from the lost frames, so the actual playout is likely to be either ahead or behind the scheduled playout by a small difference. However, this can be corrected by the adaptive playout algorithm by accordingly scaling the following frames.

Note that this method introduces a delay of one-frame time, so the playout times of the adaptive playout algorithm should be offset by one frame. However, since the complexity of this algorithm is higher than other waveform substitution techniques, it also offers better audio quality and may handle high loss rates of various patterns.

5. Implementation on the Nomadik processor

The adaptive audio playout system has been designed for the implementation on the STMicroelectronics Nomadik multimedia processor family. The Nomadik architecture [16] is specifically designed for mobile multimedia applications, e.g., it supports MPEG-4 encoding and decoding and display sizes ranging from cell phone to PDA. Fig. 3 shows the block diagram of the Nomadik heterogeneous multiprocessor architecture. The main CPU for the system is the ARM926E-JS, which concentrates on coordination and control, while most of the media functions are performed by application-specific accelerators. The Nomadik platform includes both audio and video accelerators. It uses techniques at all levels of abstraction to minimize energy consumption: distributed heterogeneous processors, efficient algorithms for motion estimation and other compute-intensive steps, data compression on the bus, embedded memory, specialized instruction sets, and aggressive power management. The Nomadik ARM core allows software to be easily ported to the platform and supports the Open Multimedia Application Platform Interface (OMAPI).

The base system for the implementation of adaptive audio playout is represented by the porting of Linphone [17] on the Nomadik platform. Linphone is a free software VoIP application released under the GNU General Public License (GPL), and it is compliant with the SIP protocol [18], which is the most popular standard for voiceover IP communications. Additionally, Linphone includes support for a large variety of codecs (G711-ulaw, G711-alaw, LPC10-15, GSM, Speex and iLBC), while SIP and RTP functionalities are provided by the oSIP [19] and oRTP [20] libraries, respectively.

The Linphone software architecture is based on a real-time scheduler that periodically resumes a program thread in charge of handling the multimedia communication. A speech transmission is, in fact, typically made up of constant size and constant duration frames that must be received and processed at regular intervals. Therefore, the Linphone scheduler is responsible for running a queue of processing filters that, in a common scenario, are represented by the reception, decoding, and playout procedures. Thus, the very first change in the Linphone architecture has been to modify this paradigm completely. Adaptive audio playout by its nature needs a time-varying playout scheduler because the duration of each frame may change accordingly to the status of the network and to the previous history of packet delays. For the same reason, we also had to modify the information passing architecture between the processing filters that previously relied on the assumption that all speech frames had the same size.

Adaptive audio playout functions have been introduced by adding two new components in the Linphone architecture as shown in Fig. 4:

- the control logic of the adaptive scheduling algorithm, that has become part of the oRTP library functions; and
- the audio time-scaling and concealment algorithm that is applied just after the speech decoding plugin.

The basic operation of the control logic of the adaptive scheduling algorithm is to set the playout time for each frame. Before the current frame can be played out, we need to decide on its length to perform the required scaling. Its length clearly depends on the current estimate of the arrival time and playout time of the next frame. If the delay of the frame in the next packet is correctly estimated, the next frame should arrive in time and be ready by the end of the playback of the current frame. A good estimation of the network delay is therefore an important component for adaptive frame scheduling. In Linphone, we modified the RTP packet reception process by inserting a hook function that collects information on the delay of all received speech packets; then, we based our estimation on the order statistics of these delays to adapt the playout schedule to the network variations in a very reactive way. Since we cannot know the real network delay without clock synchronization between sender and receiver, we assign to the first received packet a network delay equal to zero; then, the delays of successive packets are relative to this value. In our implementation, the delays are collected for a sliding window of the past *w* packets. The determination of the estimated delay for the next packet is based on the order statistic of these delays, that is, the sorted version of the delays collected over the sliding window. In statistics, the k-th order statistic of a statistical sample is equal to its k-th smallest value. It is shown [21] that, given D^k as the k-th order statistic, the expected probability that a packet with the same delay statistic can be received by D^k is $k/(w+1)$. As a consequence, given a user-specified acceptable loss rate for the application $\hat{e}$, we may look for the index $\hat{k}$ and the corresponding delay $D^{\hat{k}}$ that allows us to achieve the given loss rate with the smallest delay, i.e., the greatest D^k such that $\frac{k}{w+1} \leq 1 - e$.

The index of this delay can then be computed as $\hat{k} = \lfloor (w+1)(1-\hat{e}) \rfloor$ and the playout deadline can be approximated by the interpolation between $D^{\hat{k}}$ and $D^{\hat{k}+1}$. Thus, the playout deadline determines the scaling factor for the current packet.

Another hook function has been inserted in the playout control process of Linphone to keep a list of the actual frames present in the application playout queue updated. Thus, we can analyze the playout buffer to locate missing frames that must be concealed before the playout of the current frame. First, we check if the current frame is the right one to be played out for that time

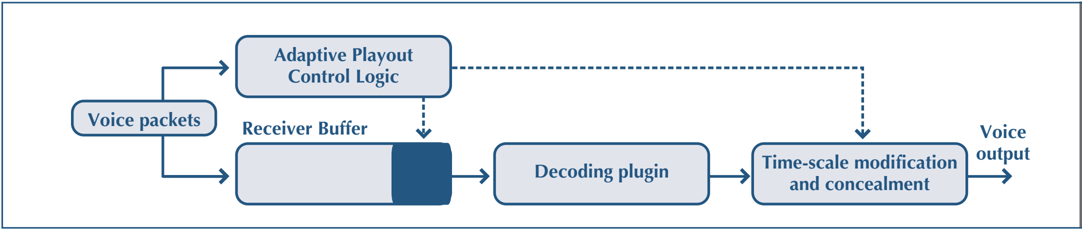
Adaptive Playout Control Logic
Voice packets
Receiver Buffer
Decoding plugin
Time-scale modification and concealment
Voice output

Figure 4: Adaptive audio playout framework.

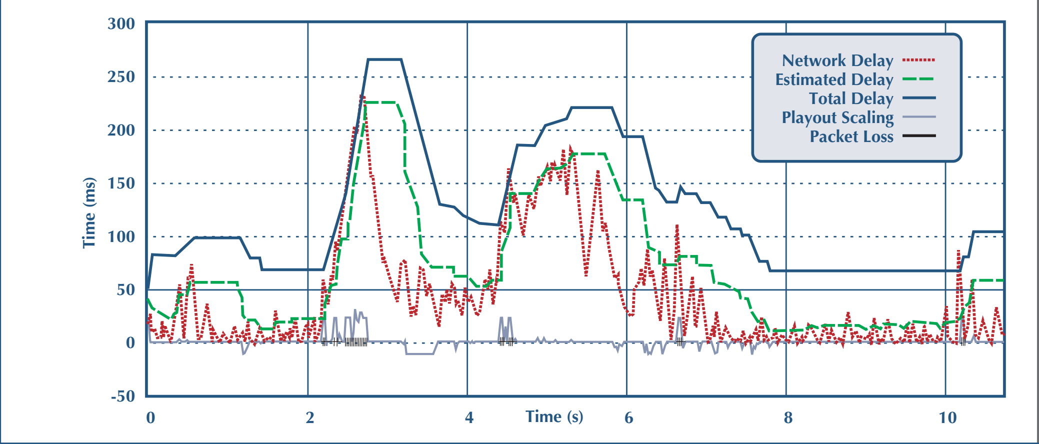

Figure 5: Example of the adaptive audio playout scheduling. As a function of the network delay (red), the algorithm estimates the expected delay of the next packet (green). When necessary, it modifies the playout deadline by a scaling factor (blue) so that the total delay experienced by the audio packets (black) may closely follow the network delay.

instant (i.e., if its RTP timestamp corresponds to the timestamp of the frame to be played out); second, we identify by the RTP timestamps if the previous and/or next frames are missing. Then, we set appropriate values for the frame time-scaling process as well as an indicator of the type of concealment to be applied and a reference to future or past speech frames, if available.

Single-packet WSOLA audio time-scaling for frame compression/expansion and loss concealment is implemented just after the speech decoding process, and it is independent of the specific voice compression algorithm used by the application. Given the decoded speech frame, the scaling factor set by the playout control logic, and the information about the availability of prior or subsequent frames, the current frame is modified to match the desired playout deadline. Depending on the actual duration of the frame after time-scaling, the Linphone processing thread is then set to be rescheduled after the same time interval.

Fig. 5 shows a summary representation of the new functionality implemented on the Nomadik architecture. For each speech packet, our algorithm estimates the expected delay of the next packet (green) as a function of the history of the network delay (i.e., arrival time) of prior packets (red). The presented time-domain time-scale audio modification algorithm is then used for changing the optimal playout deadline for the next speech frame that needs to be resized by a scaling factor (blue) that depends also on the possible loss of adjacent speech frames (blue points). Playout buffering delay is thus adapted in order to keep the total delay (black) always very close to the network delay. The algorithm flexibility allows the reduction of the average buffering delay while reducing late loss at the same time. Hence, the tradeoff between buffering delay and late loss is improved with respect to traditional buffering algorithms.

6. CONCLUSION

In this paper, we described the implementation of an adaptive playout scheduling scheme for speech communications on the STMicroelectronics Nomadik multimedia processor. These techniques allow high robustness against the loss of late packets and maintain very low buffering delay. The application may in fact dynamically adjust the receiver playout buffer size to the varying estimate of the network delay; short-term order statistics of the arrival time of each multimedia packet are used for that purpose.

As a consequence, the playout time of the voice frame must also change. To maintain proper reconstruction of continuous audio playout, each speech frame is then time-scaled according to the delay estimate using single-frame WSOLA. This algorithm operates in the time domain with acceptable computational complexity and enables time compression and expansion of speech frames without altering the audio frequency content. The same technique has also been adopted for the concealment of missing frames with the introduction of a processing delay of only one frame.

7. REFERENCES

[1] D. Cohen, "Issues in transnet packetized voice communication," in Proc. Fifth Symposium on Data Communications, September 1977, pp. 6.10-6.13.

[2] C. Weinstein and J.W. Forgie, "Experience with speech communication in packet networks," IEEE Journal on Selected Areas in Communications, vol. 6, no. 1, pp. 963-980, December 1983.

[3] Recommendation ITU-T G.114, "One way transmission time," Int'l Telecommunication Union, Geneva, 1996.

[4] R. Ramjee, J.F. Kurose, and D.F. Towsley and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio application in wide-area networks," in Proc. INFOCOM, pp. 680-688, 1994.

[5] S.B. Moon, J.F. Kurose, and D.F. Towsley, "Racket audio playout delay adjustment: performance bounds and algorithms," Multimedia Systems, vol 6, n. 1, pp. 17-28, Janunary 1998.

[6] Y.J. Liang, N. Farber, and B. Girod, "Adaptive playout scheduling using time-scale modification in packet voice communications," in Proc. ICASSP, vol. 3, May 2001, pp. 1445-1448.

[7] Fang Liu and C.-C. Jay Kuo, "Quality enhancement of packet audio with time-scale modification," in Proc. Multimedia Systems and Applications, July 2002

[8] W. Verhelst, "Overlap-Add Methods for Time-Scaling of Speech," Speech Communication, vol. 30, no. 4, pp. 207-221, 2000.

[9] Y.J. Liang, N. Farber, and B. Girod, "Adaptive Playout Scheduling and Loss Concealment for Voice Communication over IP Networks," IEEE Trans. on Multimedia, vol.5, no.4, pp. 532-543, December 2003.

[10] A.V. Oppenheim, R.W. Schafer, J.R. Buck, "Discrete-time signal processing", Prentice Hall, 1999.

[11] R.K.C. Tan and A.H.J Lin, "A Time-Scale Modification Algorithm Based on the Subband Time-Domain Technique for Broad-Band Signal Applications," Journal of AES, vol. 48, no. 5, pp. 437-449, May 2000.

[12] T.F. Quatieri, "Discrete time speech signal processing: principles and practice," Prentice Hall, October 2001.

[13] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high-quality time-scale modification of speech," in Proc. IEEE Int. Conference on Acoustics, Speech, and Signal Processing, vol.2, 1993, pp. 554-557.

[14] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod, "A new technique for audio packet loss concealment," in Proc. IEEE GLOBECOM, November 1996, pp. 48-52.

[15] G. Spleesters, W. Verhelst, and A. Wahl, "On the application of automatic waveform editing for time warping digital and analog recordings," in Proc. 96th AES Convention, preprint 3843, February 1994.

[16] A. Artieri, V. D'Alto, R. Chesson, M. Hopkins, and M.C. Rossi, "Nomadik^{TM} open multimedia platform for next-generation mobile devices," STMicroelectronics Technical Article TA305, 2003.

[17] Linphone SIP application ver. 1.1.0, http://www.linphone.org/.

[18] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, 2002.

[19] "GNU oSIP library ver. 2.2.0," http://www.gnu.org/software/osip/.

[20] "oRTP - a Real-time Transport Protocol stack under LGPL," http://www.linphone.org/ortp/.

[21] R.V. and E.A. Tanis, "Probability and Statistical Inference," Macmillan, New York, 1997.

■ **Contact: st.journal@st.com** ■