Audio Engineering Society

Convention Paper

Presented at the 127th Convention
2009 October 9–12 New York NY, USA

Performance metrics for network audio systems: methodology and comparison

Nicolas Bouillot, Mathieu Brulé, and Jeremy R. Cooperstock

Centre for Interdisciplinary Research in Music Media and Technology, McGill University, Montreal, QC, Canada

Correspondence should be addressed to Nicolas Bouillot (nicolas@cim.mcgill.ca)

ABSTRACT

Network audio transmission is becoming increasingly popular within the broadcast community, with applications to Voice over IP (VoIP) communications, audio content distribution, and radio broadcast. Issues of end-to-end latency, jitter, and overall quality, including glitches of the delivered signal, all impact on the value of the technology. Although considerable literature exists comparing audio codecs, little has been published to compare systems in terms of their real-word performance. In response, we describe methods for accurately assessing the quality of audio streams transmitted over networks. These methods are then applied to an empirical evaluation of several audio compression formats supported by different streaming engines.

1. INTRODUCTION

Existing comparison metrics for audio quality can be classified as either subjective or objective. In the former category, participants' test scores are averaged to produce the mean opinion score (MOS). An improvement over MOS is the commonly applied MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) test [1], targetted to evaluation of intermediate audio quality levels. While these methods reduce the impact of individual subjective effects on speech quality assessment, they have been criticized as overly expensive and time consuming [2]. Objective methods, in contrast, are intended to evaluate quality degradation of the delivered stream by computing parameters such as the signal-to-noise ratio (SNR), mean-squared error (MSE), cepstral or spectral distances [3]. Such automatic evaluation metrics are invaluable for applications in which a real-time quality assessment is required [4]. This approach, however, suffers from two drawbacks: First, it usually requires access to signals at both the transmitting and receiving ends, which is sometimes impractical, and second, it cannot take into account any

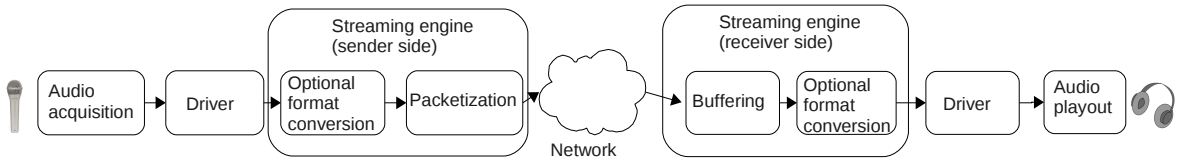


Fig. 1: Components involved during audio streaming

factors other than signal degradation, e.g., variable listening levels, echo, and delay [5]. Although many experiments have been conducted in this manner to evaluate perceptual quality of codecs using both subjective and objective techniques, little has been done in the context of end-to-end network transmission systems, for which additional factors such as robustness to network congestion, packet-delay variation (*jitter*), and error concealment are critical. For instance, in most cases and independently of the audio codec employed, an undersized queue, failing to avoid network jitter, and network losses lead to missing audio data that results in audible glitches during playout.

To address this shortcoming, we have developed an objective method for end-to-end Quality of Service (QoS) analysis for audio transmission, which includes accurate measurement of latency and jitter, as well as an evaluation of the number of glitches based only on the delivered audio data. These parameters, which have not previously been evaluated together, provide a meaningful indicator of quality for the entire transmission path, rather than being restricted to audio coding quality alone. Using our method, we analyze seven popular compressed audio streaming systems, and compare these against a reference audio transmission in which no compression was used. Coding formats included G.722, MPEG 1 Layers 2 and 3, and AAC-HE. The results allow for evaluation of the latency introduced by the streaming engine, independent of network delays, and a comparison of algorithmic delays. Our method can also be considered as a first step towards automatic, external and real-time evaluation of audio transmissions engines.

The remainder of this paper is organized as follows. In Section 2, we present a general overview of the internal components of audio streaming engines. We then introduce our methodology for measuring audio latency and delivered audio quality in Section 3. Finally, we provide experimental results in Section 4 and conclude with some observations and discussion of future work in Section 5.

2. STREAMING ENGINES

This section provides an overview of streaming engine architectures, in order to demonstrate that audio compression is not the sole determinant of the resulting acoustic quality. Rather, the quality is additionally affected by external errors described here, as well as some internal mechanisms commonly employed at the receiver for reducing effects of such errors.

Figure 1 illustrates the components involved in live audio streaming between a server and one or more clients. At both ends, the streaming engine accesses sound cards through drivers to provide audio acquisition and playout. Signal quality and introduced latency can be optimized using professional audio equipment; however, the sound card at the client is subject to clock skew relative to the server [6], which can cause data underflow or overflow. Streaming engines optionally provide a format conversion component to enable audio transmission between heterogeneous hardware, which introduces additional latency. For the purposes of this paper, we consider audio compression as a format conversion that may also introduce significant latency and quality degradation.[1]

The packetization component at the server is responsible for filling a network packet with audio data. Since data are produced periodically, packetization determines the tradeoff between introduced latency and network scheduling: a small *packet size* incurs lower latency since the packetization component

[1]Note that this is not always the case, as ultra-low-delay codecs, such as ULD [7] and CELT (`www.celt-codec.org`), which introduce less than 10 ms of latency, allow for high-fidelity audio transmission over low bandwidth networks.

waits less time before sending the current packet, but requires a higher scheduling rate or *packet rate* from the networking layer. At the receiver, the buffering component is responsible for delivering audio periodically to the optional format converter or sound driver. This is achieved by maintaining received data in an ordered queue, whose size determines the tradeoff between latency and variability in packet arrival times: a small queue size minimizes latency but may result in a disrupted audio stream if data is received late due to higher than expected jitter.

As an external factor beyond the control of the streaming engine, the available bandwidth and scheduling on IP networks are typically shared equally among clients, resulting in significant jitter and loss in congested networks.[2]. Variations in jitter and packet loss should be handled by the streaming engine's buffer manager to minimize the effect of late and/or missing audio data.

Two compatible strategies, sender-based repair or receiver-based error concealment, can be employed to reduce the perceived effect of missing audio data [8]. The former category includes both active retransmission protocols and forward error correction (FEC), the latter which passively adds redundancy to the stream. For example, the parity FEC [9], which has been standardized by the IETF Audio/Video Transport Charter, computes the parity of (typically four) successive packets to be sent and appends this to each transmitted packet. At reception, a missing packet can be recovered from the parity FEC data contained in other received packets.

Recovery of missing audio data can also be achieved purely by the receiver through *error concealment* techniques, which, in contrast with sender-based methods, do not impose additional bandwidth overhead. For transport of speech audio, Perkins *et al.* state that concealment is effective for up to 15% packet loss when packets contain between 4 and 40 ms of audio [8]. The simplest form of this technique, repair by insertion, simply replaces a missing packet by silence, white noise, or repetition of the previous packet. Although this requires minimal processing power, it introduce potentially perceptible discontinuities (*glitches)* in the audio waveform, which have been characterized in terms of their impact on the intelligibility of speech [10]. As a result, more complex concealment techniques, such as interpolation and regeneration, are favored. These provide perceptually superior results, but require more processing power. For instance, interpolation from surrounding packets using a time-scale modification involves stretching the audio on either side of the lost packet to replace the missing data [11].

[2]Note that this is a general assumption concerning end-to-end communication. Even if some sub-parts of the Internet provide guaranteed performance, this cannot be assumed for every possible IP communication.

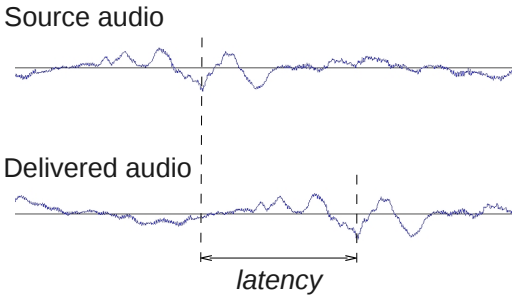


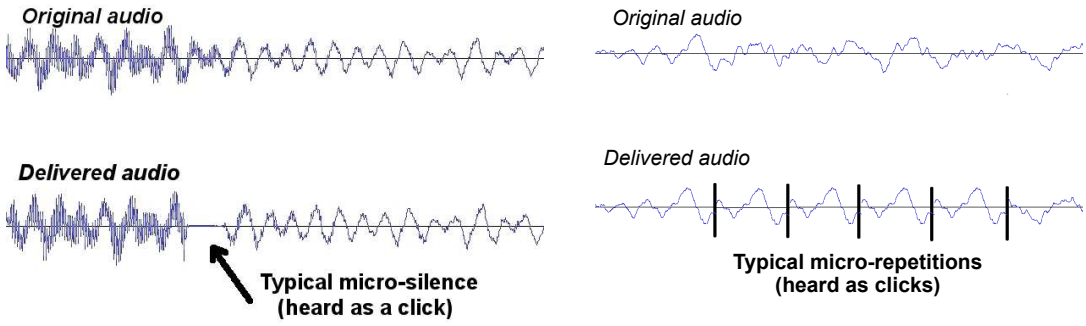
Fig. 2: End-to-end latency measurement using a multichannel editor.

3. METHODOLOGY

As described in the previous section, the resulting audio quality in not only affected by the compression codec, but also by various factors related to the audio transport. To account for the complexity of these effects, we analyze the behavior of a streaming engine based on two components: (i) a comparison of simultaneous recordings of the original and received audio streams to evaluate the effects of latency and jitter, and (ii) a glitch analysis applied only to the received stream. This approach allows for practical evaluation of streaming audio systems based simply on the audio streams themselves.

3.1. Timing analysis: latency and jitter

End-to-end latency and jitter are affected by many aspects of hardware, software and the underlying network. In order to quantify the effects on the audio stream, we simultaneously record the sound source and the received audio using a multichannel recorder. This is trivial when the transmitter and receiver are nearby. In the more general case, with a physically remote receiver, the delivered audio can



(a) Micro-silence example (b) Micro-repetition example

Fig. 3: Defects observed in delivered audio

be streamed back through the network, or failing this, separate recordings can be made with time-stamps provided by a global reference (e.g., network time protocol based on GPS). These alternatives are less desirable, however, as there is no guarantee that network performance is symmetric in a round-trip and the accuracy of network time protocols is subject to the operating system kernel.

Using the simultaneously recorded streams, we employ a simple manual method for latency evaluation, illustrated in Figure 2. The sound files are displayed and compared in a multichannel audio editor. Zooming into the waveform allows us to identify two samples representing the exact same sound in the two files, from which their temporal offset can be measured to determine latency. This method, although manual, provides measurements accurate to a single sample. Note that in order to be amenable to visual pattern recognition, the audio content must be non-periodic; we therefore avoid purely sinusoidal or other repeating signals.

Jitter is analyzed by comparing both recordings using a Dynamic Time Warping (DTW) analysis, a technique for finding an optimal alignment between two time-dependent audio sequences. For our analysis, we use MATCH [12], a DTW toolkit for audio recordings of different versions of the same content. In order to identify similar instants in the two sound files, MATCH computes a sequence of Fast Fourier Transforms with a window of 2046 audio samples (46.4 ms for audio recorded at a 44100 Hz sampling rate) overlapping by half their size (23.2 ms). Accuracy of the resulting analysis is restricted by the interval between consecutive windows, motivating us to use this analysis for jitter only.

3.2. Glitch analysis

We provide here a mechanism for analysis of streamed compressed audio, importantly, without the need for the original source. The typical cause of quality degradation is missing audio data at one of the components in the pipeline of Figure 1. When this occurs, and depending on the error recovery mechanism applied, a glitch can be heard in the playout. In earlier work on uncompressed audio transmission [13], our glitch-detection algorithm consisted of filtering delivered audio in an 8-9.5 kHz band, followed by an amplification of +25 dB. The resulting audio file exhibited spikes corresponding to the actual glitches, which were counted by an amplitude analysis. For reliability, this was applied to both the source and received audio signals, and any glitches detected in the former were ignored in the results. However, this method cannot be applied to typical compressed audio transmissions, since audio coding affects the frequency response, leading to inconsistent results at the bandpass step. This motivated the development of a new glitch-detection method based on spectral analysis, which looks for sudden and punctual variations of a frequency component in the same range of 8-9.5 kHz.

During our experiments, we observed two types of defects in the delivered audio streams, most likely

related to error concealment techniques. The first, seen in Figure 3(a), is the result of silence insertion inside an audio stream. The second is the micro-repetition, illustrated in Figure 3(b), which is likely equal in duration to one packet, but may equally be due to a similar concealment strategy elsewhere in the pipeline. We now present algorithms suitable for detection of such defects in the audio stream, without the need for the original source. Later, in Section 4, we discuss the performance of these algorithms on speech and music.

3.2.1. Detecting micro-silence

As seen in Figure 3(a) micro-silence is the result of missing audio data replaced by null values. Obviously, this can only be perceived by a listener if interleaved with sound, rather than occurring during a normal period of silence.

Input: $signal, threshold$
Output: $glitch$
$glitch, energy := 0, \infty$
for *each window* **do**
  Y:=fft($window$)
  Y:=abs(Y)
  $curenergy := \sum(Y)$
  if $curenergy > threshold \times energy$ **then**
    $glitch+ = 1$
  end
  move $window$ from its size
  $energy := curenergy$
end

Fig. 4: Algorithm for micro-silence detection

Micro-silence can be described as sudden variations of acoustic energy, dropping from high to null and then quickly reverting from null to high. Based on this observation, we designed the algorithm presented in Figure 4 for automatic detection of micro-silence. A suitable threshold of minimum energy variation, corresponding to such a glitch, must be specified; for the samples with which we experimented, a value of 25 was found, empirically, to be optimal, although we have not yet had the opportunity to verify this across a diverse set of data. An FFT-based windowed analysis allows for comparisons of energy variations across successive small time intervals.

The window size is a critical factor for reliability as we want to avoid a false positive from the sudden energy variation resulting from the onset or termination of an instrument or voice sound. Our experiments suggest that a window size of 64 samples (1.4 ms at a sampled rate of 44100 Hz) is a sufficiently brief duration, less than the typical few milliseconds of a sound attack time.

3.2.2. Detecting micro-repetition

Micro-repetition, as seen in Figure 3(b), results from the repetition of small intervals of the audio signal. Assuming a constant duration for the repeated interval within a particular session,[3] the detection of a micro-repetition becomes a pattern recognition problem.

Input: $signal, glitchsample, threshold$
Output: $glitch$
$refenv$:=envelope($glitchsample$)
for *each window* **do**
  $winenv$:=envelope($window$)
  if *corr(*$refenv, winenv$*)*$> threshold$ **then**
    $glitch+ = 1$
  end
  move $window$ from 1 sample
end

Fig. 5: Algorithm for micro-repetition detection

We designed our detection algorithm, presented in Figure 5, applying the *linear correlation coefficient*, which is widely used for measuring associations between sampled data. This requires an extract of the signal to detect, denoted *glitchsample* in the algorithm. Obviously, the extract must contain at least one repetition. At initialization, the envelope of this extract, called $refenv$, is computed. An analysis window of equal size to *glitchsample* is moved through the data, its envelope computed and correlated with the extract to determine the linear correlation coefficient. A threshold is then used for determining whether the analyzed instant is detected as a glitch.

[3] This is a reasonable hypothesis in the case of packet repetition for error concealment. The use of fixed packet size is widely adopted in audio transmission protocols, such as in the Real-time Transmission Protocol and related standards.

4. EXPERIMENTAL RESULTS

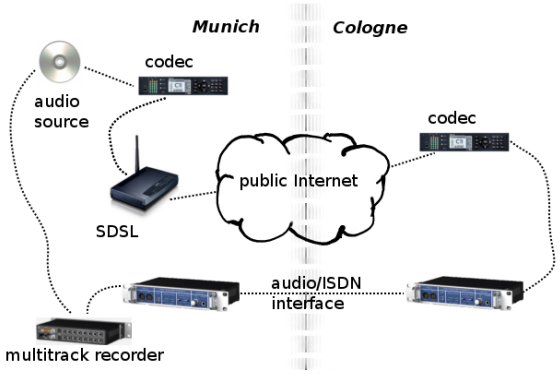


Fig. 6: Testbed architecture deployed in Munich and Cologne.

Our testbed, illustrated in Figure 6, which was set up during the AES 126th Convention in Munich, consisted of five commercial encoders in Munich and their corresponding decoders in Cologne. The source audio signal was provided to the encoders from a single channel of a compact disk recording (44.1 kHz sampling rate at 16 bits) of approximately 65 s of speech and 80 s of music. Session initiation between encoder and decoder was established via a public SIP server and the encoded data transferred to Cologne over a symmetric 2 Mbps DSL connection to the public Internet. The decoders were connected through a mixing console in Cologne to an ISDN downlink, allowing for recovery of the decoded signal with no further compression and the introduction of minimal latency. In Munich, the decoded signal was then played out and recorded simultaneously with the source signal using a multichannel recorder. The same audio source signal was used for each test.

In addition, the performance of a Luci Live codec, running on a Windows mobile enabled telephone, was tested with transport exclusively over a wireless network.

4.1. Time analysis

Timing analyses are presented in Figure 7. Interestingly, for a given compression format, performance varies between systems. Furthermore, observed end-to-end latency l_{ete} is significantly greater than the sum of the corresponding algorithmic delay, listed in Table 1 and the round-trip ISDN reference delay, l_{ISDN}, estimated in the order of 42 ms.[4] We estimate the engine latency, l, which includes coding, buffering, and decoding, but excludes network transport delay, by averaging several measurements of:

$$l = l_{ete} - l_{ISDN}$$

Comparing the ratios of average engine latencies with the corresponding algorithmic delay indicates that time spent on coding, decoding, and network transport represents only a small part of the overall end-to-end latency. This suggests the dominance of buffer size, rather than choice of encoding algorithm, as a determinant of end-to-end latency. Moreover, the results indicate significant differences of implementation or configuration among the various engines.

Jitter analysis was performed as described in Section 3.1, using metadata extracted from MATCH. Significant jitter was observed two transmissions in the speech set and one in the music set, with a maximum of 100 ms. As the jitter occurred suddenly during periods of silence, this suggests either the use of adaptive jitter management or possibly, voice activity detection (VAD) for adapting audio playout during periods of silence.

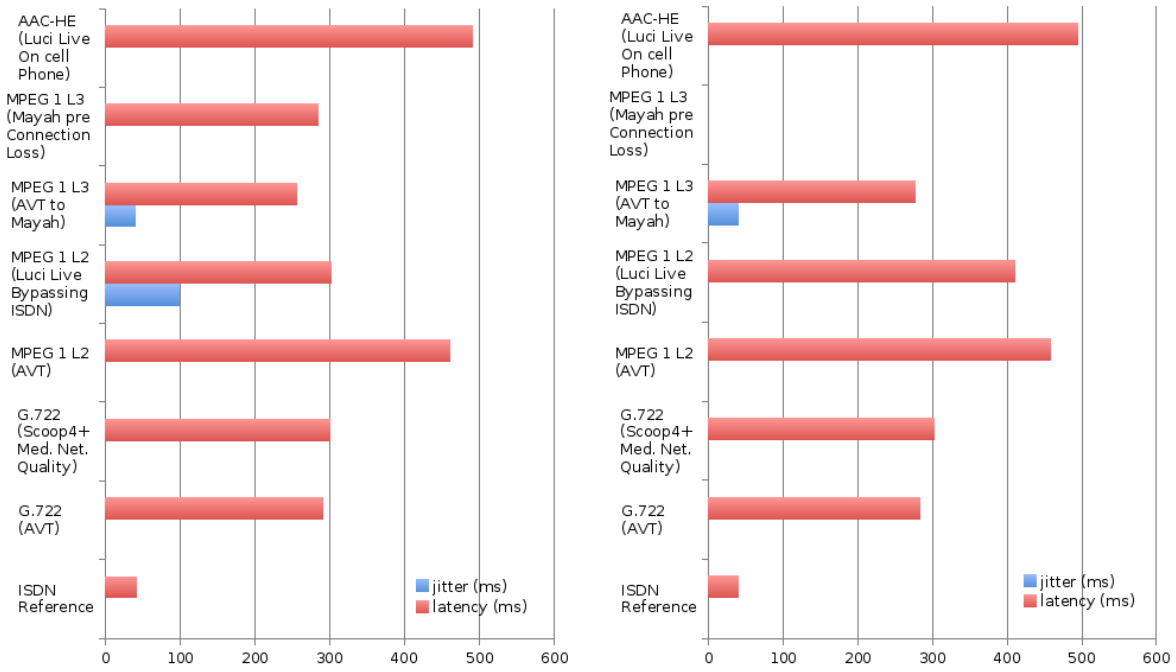
4.2. Glitch analysis

Engine	codec	defect	speech	music
AVT to Mayah	MP3	silence	3	3
AVT	MP2	repetition	2	9

Table 2: Number of glitches measured for speech and music sets. No glitches were observed in other sessions. The AVT to Mayah engine refers to the condition in which an AVT encoder and Mayah decoder were used together. MP2 and MP3 refer to MPEG-1 Audio Layer 2 and MPEG-1 Audio Layer 3, respectively.

The glitch analysis algorithms presented in Section 3.2 were implemented in MATLAB, and applied to the recorded samples of audio transmissions.

[4] Unfortunately, a separate latency measurement was not made through the DSL link.



(a) Speech set (b) Music set

Fig. 7: Time analysis

The results are presented in Table 2 and were verified manually by multiple listenings to the extracts. Consistent with our manual observations, the algorithm for micro-silence detection founds glitches in the AVT-to-Mayah transmission of MPEG 1 L3 only. Two glitches were ignored by the algorithm as they occurred during periods of relatively low sound amplitude.

Our second algorithm detected micro-repetition in the AVT transmission of MPEG 1 L2 only. Automatic detection using a *threshold* of 0.51 was consistent with manual observations for music, with our algorithm identifying all nine glitches at the correct positions, but provided three false positive detections for speech samples.

5. CONCLUSION

Whereas existing metrics for assessing network audio systems evaluate only the perceptual quality of the underlying codecs, we investigated audio quality over the entire transmission system. This was enabled by our design of a new objective methodology, which considers additional metrics of end-to-end latency, jitter and the number of glitches. This methodology was tested on seven commercial compressed audio streaming systems and compared against a reference audio transmission in which no compression was used. Results indicated that codec latencies represent only a small part of the entire system latency.

A further contribution of this paper is the demonstration of glitch analysis techniques that can be applied to received audio signals without requiring access to the original source audio. This analysis was applied to received samples of speech and music, for which we observed the well-known defects of micro-silence and micro-repetition, commonly associated with error concealment techniques.

Our ongoing efforts are focused on improving the reliability of our automatic glitch analysis, particularly for speech, for which some vocal inflections presently lead to occasional false positives. We believe that improvements in this area will enable automatic external monitoring and possibly assist in automatic

codec	algorithmic latency (ms)	average engine latency (ms)	ratio
G.722	< 1	254	254
MPEG-1 L2	> 45	340	7.4
MPEG-1 L3	> 80	228	2.8
AAC-HE	> 110	450	4.08

Table 1: Comparison between algorithmic delay and measured delay. Algorithmic delay values are taken from Hans-Heinrich Hansen *et al.* [14].

configuration of audio transmission systems.

6. ACKNOWLEDGEMENT

The authors would like to thank Mitchel Benovoy for his valuable comments and suggestions for the design of the micro-repetition algorithm and Christian Diehl for pointing us to documentation regarding algorithmic codec latencies. Special thanks are due to Heinz Peter Reykers for organizing the workshop on New Technologies for Audio over IP and inviting our participation in this event.

7. REFERENCES

[1] "Method for the subjective assessment of intermediate quality level of coding systems (mushra).." ITU-R Recommendation BS.1534, 2003.

[2] G. Bucci, F. Franciosi, and P. Valocchi, "The measurement of audio-codec sound quality," in *Instrumentation and Measurement Technology Conference, 1996. IMTC-96. Conference Proceedings. 'Quality Measurements: The Indispensable Bridge between Theory and Reality'., IEEE*, vol. 1, pp. 622–627 vol.1, 1996.

[3] M. Guéguin, R. Le Bouquin-Jeannès, V. Gautier-Turbin, G. Faucon, and V. Barriac, "On the evaluation of the conversational speech quality in telecommunications," *EURASIP J. Adv. Signal Process*, vol. 2008, pp. 1–15, 2008.

[4] "Method for objective measurements of perceived audio quality." ITU-R Recommendation BS.1387-1, 2001.

[5] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei, "Quantifying skype user satisfaction," in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 399–410, ACM, 2006.

[6] E. Brandt and R. Dannenberg, "Time in distributed real-time systems," in *Proc. of the 1999 International Computer Music Conference*, (San Francisco), pp. pp. 523–526, 1999.

[7] S. Wabnik, G. Schuller, J. Hirschfeld, and U. Krämer, "Reduced bit rate ultra low delay audio coding," in *Proceedings of the 120th AES Convention*, May 2006.

[8] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet-loss recovery techniques for streaming audio," *IEEE Network Magazine*, Sept./Oct. 1998.

[9] J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction." Network Working Group Request for Comments: RFC 2733, December 1999.

[10] G. Miller and J. Licklider, "The intelligibility of interrupted speech." Journal of the Acoustic Society of America, 22: p.167-173, 1950.

[11] H. Sanneck, A. Stenger, K. Younes, and B. Girod, "A new technique for audio packet loss concealment." Proceedings IEEE Global Internet 1996 (Jon Crowcroft and Henning Schulzrinne, eds.), pp. 48–52, (London, England), 1996.

[12] S. Dixon and G. Widmer, "Match: A music alignment tool chest," in *in Proc. ISMIR*, 2005.

[13] N. Bouillot and J. R. Cooperstock, "Challenges and performance of high-fidelity audio streaming for interactive performances," in *9th Intl.*

Conference on New Interfaces for Musical Expression, (Pittsburgh, USA), 2009.

[14] H.-H. Hansen, C. Diehl, U. Andersen, J. Simon, W. Ludwig, and J. R. D. Wiese, "Audio-via-ip compendium, basic principles, practices and applications." Published by: MAYAH Communications GmbH, 2007.