Technical Report WSI-2009-3

Considering Bluetooth's Subband Codec (SBC) for Wideband Speech and Audio on the Internet

Christian Hoene, Mansoor Hyder

October 2009

Interactive Communication Systems
Wilhelm-Schickard-Institut
Universität Tübingen
Sand 13
D-72076 Tübingen, Germany
http://www.net.uni-tuebingen.de/

ISSN 0946-3852

Abstract

The Bluetooth Special Interest Group (SIG) has standardized the subband coding (SBC) audio codec to connect headphones via wireless Bluetooth links. SBC compresses audio at high fidelity while having an ultra-low algorithm delay. To make SBC suitable for the Internet, we extend it by using a time and packet loss concealment (PLC) algorithm that is based on ITU's G.711 Appendix I. The design is novel in the aspect of the interface between codec and speech receiver. We developed a new approach on how to distribute the functionality of a speech receiver between codec and application. Our approach leads to easier implementations of high quality VoIP applications.

We conducted subjective and objective listening tests of the audio quality of SBC and PLC in order to determine an optimal coding mode and the trade-off between coding mode and packet loss rate. More precisely, we conducted MUSHRA listening tests for selected sample items. These tests results are then compared with the results of multiple objective assessment algorithms (ITU P.862 PESQ, ITU BS.1387-1 PEAQ, Creusere's algorithm). We found out that a combination of the PEAQ basic and advanced values best matches—after third order linear regression—the subjective MUSHRA results . The linear regression has coefficient of determination of $R^2 = 0.907^2$. By comparison, our individual human ratings show a correlation of about $R = 0.9$ compared to our averaged human rating results.

Using the combination of both PEAQ algorithms, we calculate hundred thousands of objective audio quality ratings varying audio content and algorithmic parameters of SBC and PLC. The results show which set of parameters value are best suitable for a bandwidth and delay constrained link. The transmission quality of SBC is enhanced significantly by selecting optimal encoding parameters as compared to the default parameter sets given in the standard.

Finally, we present preliminary objective tests results on the comparison of the audio codecs SBC, CELT, APT-X and ULD coding speech and audio transmission. They all allow a mono and stereo transmission of music at ultra-low coding delays (<10ms), which is especially useful for distributed ensemble performances over the Internet.

Contents

List of Tables

List of Figures

1 Introduction

The motivation of this work is based on the observation that the quality of telephony has hardly improved over the last few decades. It appears that the speech quality has also suffered the same fate and it seems in even worse condition.

Traditionally, the quality level of a PSTN call should achieve a "toll quality", which is the least quality for which customers are willing to pay. A call has toll quality if it has a speech quality with a MOS-LQS value above 3.8 or an E-Model R-factor rating of 70 or above. Toll quality can be achieved with a narrow bandwidth of up to 3400 Hz. If two persons speak together over the phone, a delay between 50 and 150 ms provides a good conversation quality [4], because usually one person speaks after the other interactively.

In the days of broadband Internet access and toll-free telephone, there is no reason anymore to limit the quality of calls to the toll quality—Internet calls are available for free—but to use the available bandwidth entirely to offer superb quality, if possible. Superb quality is required if—for example—musicians make music together. Then they communicate simultaneously not just interactive. If two persons speak together over the phone, a delay between 50 and 150 ms provides a good conversation quality [4], because usually one person speaks after the other interactively. If they communicate simultaneously, they notice such large delays easily. Typically, musicians communicate simultaneously, if they make music together. Then, each musician plays or sings at the same time and need to synchronise her/himself to the other musicians. Therefore the latency requirements are much harder than for interactive usages.

Research studies have shown that if two musicians are placed more than 10 meters apart, they can stay synchronized only with difficulty. At 10 m distance, the acoustic delay is - due to the speed of sound - about 25 ms. Empirical studies show that if musicians want to play together, the optimal acoustic latency should be around 11.5 ms [5, 6, 7][1]. Also, musicians demand for a very high acoustic quality.

Because of these stringent requirements, we see ensemble performing over the network as the most demanding usage scenario for the good old telephone. It can be seen as the upper limit on a quality scale which ranges from toll quality up to high fidelity and ultra low delay acoustic transmission[2].

Keeping these requirements in mind, we looked for an audio codec capable of transmitting both speech and music at low coding rates and an algorithm delay of less than 11.5 ms. We found one codec in the standards of the Bluetooth Special Internet Group (SIG). In May 2003, the Bluetooth SIG, the standardization body for Bluetooth related technologies, published a specification to support high quality audio distribution to Bluetooth devices called A2DP [1]. It

[1] If the latency becomes larger, such as in an orchestra, the musicians need to be synchronized by a conductor, which synchronizes the musicians visually.

[2] Higher quality might only be achieved if one considers binaural acoustic transmission. Refer to [8] for further information.

is intended to connect wireless headsets and headphones via Bluetooth to an audio source. With this Bluetooth profile, the wireless headsets connected to a mobile can be used to listen to the music in addition to transmitting and receiving narrow speech.

The first product supporting A2DP came in the market by the end of 2004. Major operating systems support it since 2007. Many modern mobile phones, notebooks and wireless headphones support A2DP proving that A2DP profile is a successful technology. The A2DP profile defines which audio codecs should be used over a Bluetooth link. The codecs include MPEG-1, 2 audio (MP3), MPEG-2, 4 AAC, ATRAC and the mandatory SubBand Codec (SBC). The SBC is based on an earlier work by de Bont [1] and Bernard et al. [9]. It is free to be used with all Bluetooth devices. The SBC audio codecs has a low algorithmic complexity and can be implemented for very low power devices. Thus, it is especially useful for mobile headsets and headphones, which benefit from a light battery with low energy capacity. In addition, the SBC comes with a couple of properties which make it worthwhile to consider it beyond its original usage scenario of connecting wireless headphones.

First, one of the nice features of SBC is that it is configurable to a large extend. Most encoding parameters such as the sampling rate, the number of frequency bands it compresses, the bit rate and frame size can be freely selected at run-time to cope with changed requirements. For example, if only speech has to be transmitted, its bit rate can be reduced, or even the bit rate might be further reduced during silence period. Rate reductions help to save energy in case if it is required. Also, on the Internet, if the bandwidth is too low, then both frame and bit rate can be changed.

Another feature of SBC is the algorithm delay, the delay required to encode and the delay of the audio signal, which is in the order of a few milliseconds. Thus, the SBC codecs, besides being good for high fidelity audio can be used for musician playing over the Internet.

We describe the SBC algorithm in Section 2. Because SBC does not include a concealment algorithm, in Section 3 we present the full audio band version of the reverse order replicated pitch periods (RORPP) algorithm [3] to conceal packet losses and the negative effect of playout rescheduling. Section 4 contains subjective MUSHRA and objective results on the audio quality of SBC and our PLC algorithm. We also compared the subjective MUSHRA ratings and the objective rating of six different algorithms. More precisely, first we measured the audio quality with the formal MUSHRA tests by asking for the judgement of 11 persons, collecting 646 quality judgements. Second, we measured the speech and audio quality in the instrumental methods standardized in ITU-R BS.1387-1 [10], ITU-T P.862 [11], and the algorithm described by Creusere [12]. A combination of the ITU BS.1387-1 basic and advanced versions shows the highest correlation to subjective MUSHRA rating—after applying a third order mapping function.

In Section 5 we analysed SBC's coding performances under various operational parameters. Also, we present the objective assessment results of our PLC algorithm. Finally, we compare SBC with the few other audio codecs that support the compression of audio signal at very low algorithmic delays, names namely the ultra-low delay (ULD) codec [13, 14], the APT-X codec[15], and the Constrained-Energy Lapped Transform (CELT) [16].

With the results of this work, SBC can now be optimally used for audio and wide speech transmissions and for variable bit- and frame rate transmission over the Internet. They also show that using the optimal coding mode is as important as using a good performing codec.

2 The A2DP Subband Codec

The Bluetooth's Low Complexity Subband Coding (SBC) is an audio coding system specially designed for Bluetooth (AV) applications to obtain high quality audio at medium bit rates and having a low computational complexity. It is defined in A2DP specification version 1.0 [1][1] and is based on work of Frans de Bont [17] and Rault J. Bernard et al. [9]. The specification of SBC was included to ensure interoperability of devices supporting the A2DP profile. Beside SBC, A2DP devices may support different vendor specific codecs such as MP3, AAC and ATRAC. However, only SBC is mandatory.

2.1 Functional Description

The SBC encoders take as input signed 16-bit PCM coded audio signals having a *sampling frequency* f_s of 16, 32, 44.1 or 48 kHz. SBC can run in a one channel *mono* mode or in the two channel *stereo*, *joint-stereo* or *dual channel* modes.

The SBC encoder consists of a polyphase analysis unit, a quantization unit, an Adaptive PCM coder and the final bitstream packing (Figure 2.1).

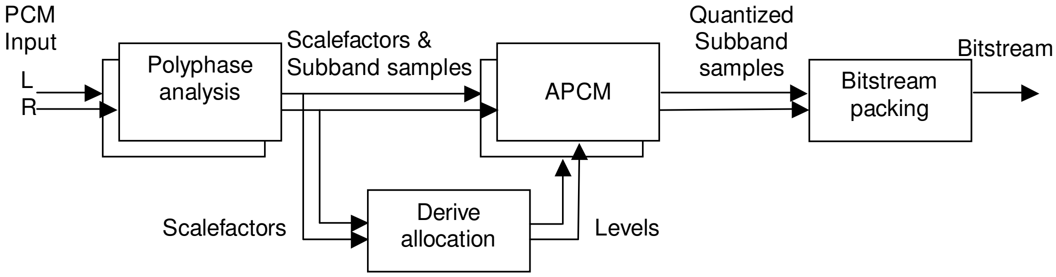


Figure 2.1: Diagram of the SBC encoder

First, the SBC encoder converts the stereo audio signal into multiple subbands which are equally spaced. The subband coding has been inspired by the similar process in MP3 encoders described in ISO/IEC 11172-3 [18] but SBC uses 4 or 8 subbands as compared to 32 subbands in MP3 and polyphase quadrature filters [19] having a size of $10 * subbands$ (instead of a size of 512). The polyphase quadrature filters convert $n = subbands$ audio sample into n single subband samples. These n samples form one *block*.

SBC collects 4, 8, 12 or 16 blocks before using these blocks to calculate the maximal loudness of each subband. The loudnesses are then rounded up the next power of two. Using scale factors,

[1] The more recent version 1.2 has a couple of editorial errors and thus is incomplete.

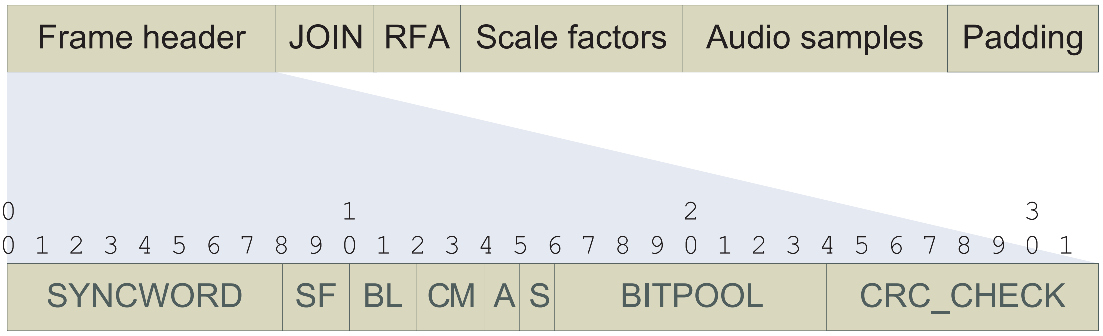


Figure 2.2: SBC frame format (SYNCWORD: Always 8 bit set to 0x9C, SF: 2 bits for sampling frequency, BL: 2 bits for number of blocks, CM: 2 bits for the channel mode, A: allocation mode, S: number of subbands, BITPOOL: 8 bits for number of bits used for the bit allocation mode, CRC CHECK: a CRC8 check over all bits of the frame header expect the SYNCWORD and all scale factors, JOIN: subband-1 bits (only in the joint stereo mode) to indicate, whether to decode the stereo channels jointly, RFA: one bit (available in the joint stereo mode) always equal to zero.

the subband audio signals are normalized to values ranging between $[-1;1]$. The normalized subband samples are not transmitted in full resolution but are quantized.

SBC supports two different algorithms for calculating how many bits should be allocated to each subband. The two modes are called SNR and LOUDNESS. The SNR mode is simple and calculates the number of bits needed, using $(\log_2 scalefactor) - 1$. The LOUDNESS mode calculates the bit needed similar to the SNR mode but it uses a weighting based on subband positions and the sampling rate. More bits are allocated to the lowest band whereas the highest bands require a lower number of bits. Also, subbands with a medium loudness are getting more bits at the costs of quiet bands.

If the requested number of bits is calculated, a limited number of bits are distributed to the band. Typically, the number of bits given the *bitpool* parameter is constant. These bits are distributed amount all subbands. The bits from a given *bit pool* are distributed in proportion to the relative number of demanded bits. Subbands that need more bits, get more bits but not necessarily all the bits they have requested for.

The SNR and LOUDNESS bit allocation run once in the mono mode and twice in the dual channel mode. The dual channel mode uses twice the number of bits given in the *bitpool* variable. In the stereo mode the bits are jointly distributed between the two channels. In the joint stereo mode, it can be decided for each subband channel whether they will be encoded as two separate channels or they will be converted to mid and side channels calculated by summing up the right and left.

Finally, the parameters describing the coding modes (Table 2.1), the scale factors and the quantized subband audio samples are packed into one frame (Figure 2.2), which is then transmitted.

On the receiving side (Figure 2.3), the decoder first calculates the bit allocation based on the received scale factors then the subband samples are reconstructed and fed into the reverse polyphase quadrature filter which generates the broadband PCM audio signal.

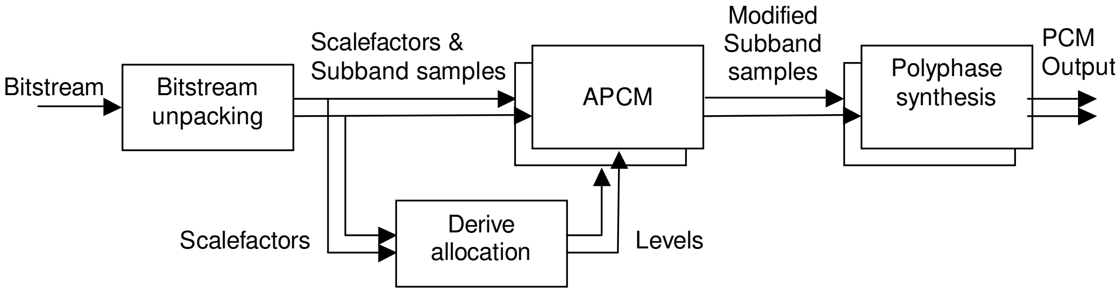


Figure 2.3: Diagram of SBC decoder [1]

Table 2.1: Description of SBC's coding parameters

Name	Range	Description
sampling frequency	16k, 32k, 44.1k or 48k	The sampling frequency at which SBC is operating.
channel mode	mono, dual channel, stereo, or joint stereo	The number of channels to be used (MONO has one channel, all others 2) and the way the two channels are compressed (jointly in STEREO and JOINT_STEREO, separately in DUAL_CHANNELS), and whether individual frequency bands can be identical (JOINT_STEREO)
bitpool	2 to 250	Expresses how many bits per audio segment (block) are used. In case of STEREO and JOINT_STEREO the bitpool is used for both channels jointly. In case of the DUAL_CHANNEL twice the number of bits is used, for each channel the amount specified in this parameter.
allocation method	SNR and LOUDNESS	Select how the bits shall be distributed on the frequency subbands. If LOUDNESS is given, they are distributed according the relative loudness of each band. If SNR is selected, instead of the loudness of the signal to noise ratio is used.
blocks	4, 8, 12, 16	This parameter controls the number of blocks which are put together in one frame. The if more blocks are transmitted in one frame, the efficiency increases at the costs of algorithmic delay and transient behaviour.

The A2DP specification allows the selection of most coding parameters freely. It only recommends eight sets of the parameters (Table 2.2).

Depending on the SBC coding parameters (Table 2.1) the length of an SBC frame, the coding rate, the frame rate and the algorithmic delay varies. The A2DP specification contains the following equation, which calculates the frame length and the bit rate. The length of frames (in

Table 2.2: Recommended sets of SBC encoder settings (with block length=16m, allocation method=LOUDNESS, subbands=8)

	Middle Quality				**High Quality**			
Channels	**mono**		**joint stereo**		**mono**		**joint stereo**	
Sampling frequency (kHz)	44.1	48	44.1	48	44.1	48	44.1	48
Bitpool value	19	18	35	33	31	29	53	51
resulting frame length (bytes)	46	44	83	79	70	66	119	115
resulting frame rate (Hz)	344.53	375	344.53	375	344.53	375	344.53	375
resulting bit rate (kb/s)	126.8	132.0	228.8	237.0	192.9	198.0	328.0	345.0
resulting algorithmic delay	4.67	4.29	4.67	4.29	4.67	4.29	4.67	4.29

bytes) are calculated as

$$framelength = 4 + \frac{subbands*channels}{2} + \begin{cases} \left\lceil \frac{blocks*channels*bitpool}{8} \right\rceil & \text{if mono or dual channel mode} \\ \left\lceil \frac{subbands+blocks*bitpool}{8} \right\rceil & \text{if joint stereo mode} \\ \left\lceil \frac{blocks*bitpool}{8} \right\rceil & \text{if stereo mode} \end{cases} \tag{2.1}$$

The bit rate in bps is determined as

$$bitrate = \frac{8 * framelength * f_s}{subbands * blocks} \tag{2.2}$$

and the frame rate in Hz as

$$framerate = \frac{f_s}{subbands * blocks} \tag{2.3}$$

The SBC's algorithmic delay is due to the encoder which reads $blocks * subbands$ samples and introduces a delay of $blocks * subbands - 1$ samples. The analysis and synthesis filters add a delay of of $10 * subbands - 1$ samples. Thus, the total algorithmic delay is calculated as:

$$delay = \frac{((blocks + 10) * subbands - 2)}{f_s} \tag{2.4}$$

2.2 Implementation

The appendix of the A2DP specifications contains the C source code of the SBC. Thus, SBC can be implemented quite easily. In addition, Bluetooth SIG provides a reference implementation with which other implementations such as our own can be crosschecked. However, because SBC is frequently used in mobile, battery powered devices considerable effort has been invested in reducing the complexity of the algorithm. For example, Hermann et al. describe a low-power implementation of SBC in [20]. Also, SBC has been implemented for BlueZ Bluetooth support of Linux kernel [21]. This implementation is available as open-source under the GPL license. In the preparation of this technical report, we found that the SBC Linux implementation has a number of bugs, resulting in poor audio quality. We used the performance tests described in [22] to find these errors.

Despite the fact that SBC is available as open source and compiled for most Linux systems, its intellectual properties rights are protected by at least one patent [9]. Because of a contract between the patent owners and Bluetooth SIG, the usage of SBC is license free for all Bluetooth devices. However, it is not publicly known which devices are covered by this contract. The patent owners were reluctant to provide precise information in which kind of applications SBC can be used without the requirement of paying license. Luckily, the patent is going to timeout in 2010.

3 Supporting Time and Loss Concealment in a Common Decoding, Concealment, and Dejittering Unit (CDCD)

The A2DP does not describe any packet loss concealment algorithm for the SBC. However, in case of transmission over the Internet, frame losses do occur. To ensure an acceptable audio quality even in the case of frame losses packet loss concealment algorithms are required. In addition, because jitter might occur due to varying packet queuing delays or MAC layer retransmissions, time concealment should be supported to slow down or speed up the playout of the audio signals.

3.1 Speech Receiver for VoIP

The traditional interface between VoIP application and speech codec used to be quite simple: The VoIP application gave audio samples to the encoder which compressed the audio signal and generated a speech frame. Also, the VoIP application received VoIP packets, took care to dejitter the frames and give the decoder speech frame (or a loss indication), which then generates the audio signal to be played out. Modern speech receivers require a sophisticated jitter buffer management which controls the decoding of speech frames. For example, the 3GPP TS 26.114 specification [2] describes a speech receiver for high quality VoIP transmission. 3GPP speech receiver enhances the classic codec design of AMR by a couple of features to make it work in an IP network. Beside an AMR narrow- and wide-band codec, it includes a jitter-buffer, supports a source-controlled and non-source-controlled rate operation, a sorted playout buffer, a jitter-buffer and clock-drift management.

We reproduce 3GPP's exemplary structure of speech receiver in (Figure 3.1), which contains a "network analyzer" and "adaptation control logic", which controls the size of the buffer, and "speech decoder" and "adaptation unit" provide the media processing functionality. More precisely, the jitter buffer unpacks the incoming RTP payloads and stores the received speech frames in a sorted order. The buffer provides the frames to the speech decoder which is a standard AMR or AMR-WB including loss concealment. The network analysis block monitors incoming packets and collects reception statistics (e.g. jitter, packet loss) that are needed for jitter buffer adaptation. The adaptation control logic adjusts the playback delay dynamically, based on the buffer status (e.g. average buffering delay, buffer occupancy, etc.), and the input from the network analyser. The adaptation control may change the time of playout even during active speech with the help of the adaptation unit. The adaptation unit shortens or extends the output signal length to allow playout delay adjustment without a large perceptual distortion. The standard suggests that “the adaptation is performed using the frame based or sample based time

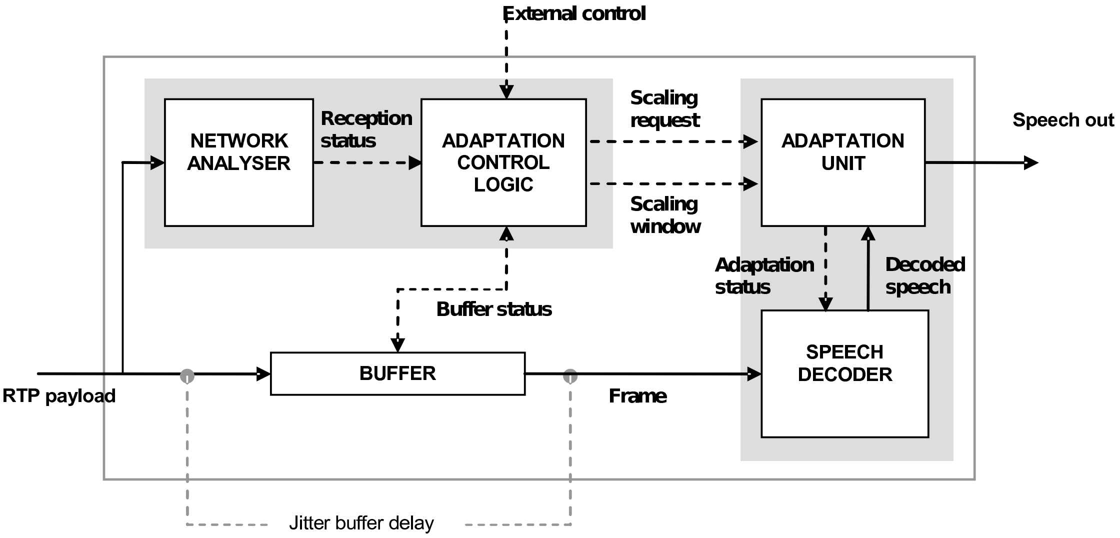


Figure 3.1: Design of the 3GPP's speech receiver [2].

scaling on the decoder output signal during comfort noise periods only or during active speech and comfort noise. ... The adaptation unit may be implemented either in a separate entity from the speech decoder or embedded within the decoder." This so called time concealment deals with the fact that low-delay de-jitter buffers adapt their play out time to the jitter introduce by the network [23, 24]. Changing of playout time can be done most easily during silence as it cannot be heard [25, 26]. Changes during an active signal are more difficult requiring algorithms such as WSOLA [27, 28], which introduce an additional algorithmic latency.

In addition, the 3GPP TS 26.114 specification suggests the use of adaptive coding rates, variable packetization by changing the number of frames per RTP packet, and the use of Forward Error Correction to cope with congestion and packet loss. The 3GPP's speech receiver add function needed for IP transmissions to speech codecs that have been developed for circuit switched speech transmission on the GSM and UMTS wireless access networks. Similar approaches and techniques are used in other modern VoIP applications such as Skype [29].

3.2 Interfacing an Internet-Codec

In 3GPP's design of a speech receiver, the AMR codecs are used because the need to support circuit switched GSM and UMTS wireless access network. In our design we have an advantage that the circuit switched operation support is not needed. Thus, we can assume that an audio codec must not support circuit switched network and can be optimized for packet based transmissions in a clean slate approach. Thus, our design can be made more lightweight.

The 3GPP TS 26.114 receiver includes beside the decoders also other signal processing tasks such as the concealment of playout time adjustments. Ideally, this task should be merged with the codec in order to simplify the overall design and enhances its performance because concealment of frame loss and the shortening and extending of the output signal are very similar tasks.

Also, at the sender, sometimes multiple frames are put into one packet to reduce the packet rate and enhance transmission efficiency. An encoder optimized for the Internet might take advantage of the packetization process because it can be configured to do the packetization by itself. Depending on the kind of encoding algorithm used, the encoder can take advantage of the higher available algorithmic delay. For example, redundancy between frames can be omitted if those frames are within a packet.

Similar the codec can optimize the FEC that is required to cope with an expected loss rate. Typically, a codec specific FEC is more efficient than a general support of FEC on a frame level.

In order to support those features and in order reduce the complexity of the speech receiver we propose a new interface in the block of a speech receiver. More precisely, in order to support time and loss concealment, we propose the design of a speech receiver which includes decoding, concealing and dejittering. Consequently, we call this design common decoding, concealment and dejittering (CDCD). Our design has two interfaces, one is for network and the other one is for audio sink. Figure 3.2 shows the interface in an event-based notation following SDL style [30]. It works as follows:

- After receiving a packet, the CDCD stores the packet. More precisely, the speech frames in the VoIP packet are kept in a sorted buffer (Function A).

- The CDCD gets an event from the audio sink requesting a block of audio samples. The CDCD must provide a block of audio sample within a limited time frame (typically a couple of milliseconds). Typically, a frame is then decoded or concealed.
  Depending on the available received frames, the CDCD can distinguish from three cases (Function B):

  1. *The next frame is available.*
     Then, the decoder decodes the frame and generates the corresponding block of audio samples (Function 1).
     - Typically, these audio samples are given to the audio sink.
     - Alternatively, if the dejittering unit may decide to speed up the play out of the audio signal, then, for example, it can skip a period of silence or drop a pitch period (Function 4).
  2. *Neither the next nor the after next frame is available.*
     At this point of time the CDCD cannot distinguish between a lost frame and a delayed frame because it doesn't know whether the next frame will still arrive or it has been lost. We assume here that the speech receiver cannot predict the process of packet transmission in the network because of lack of local knowledge. It also-doesn't know that how likely the loss or delayed event will be because of the same reasons. Then, the speech receiver cannot distinguish between a loss and a delayed packet.
     Typically, it is better to delay the playout a bit and wait for a late packet than just concealing it [31]. Thus, the CDCD should do an extrapolation of the last audio signal (Function 2) and decide later on whether to delay the playout or to conceal a frame.

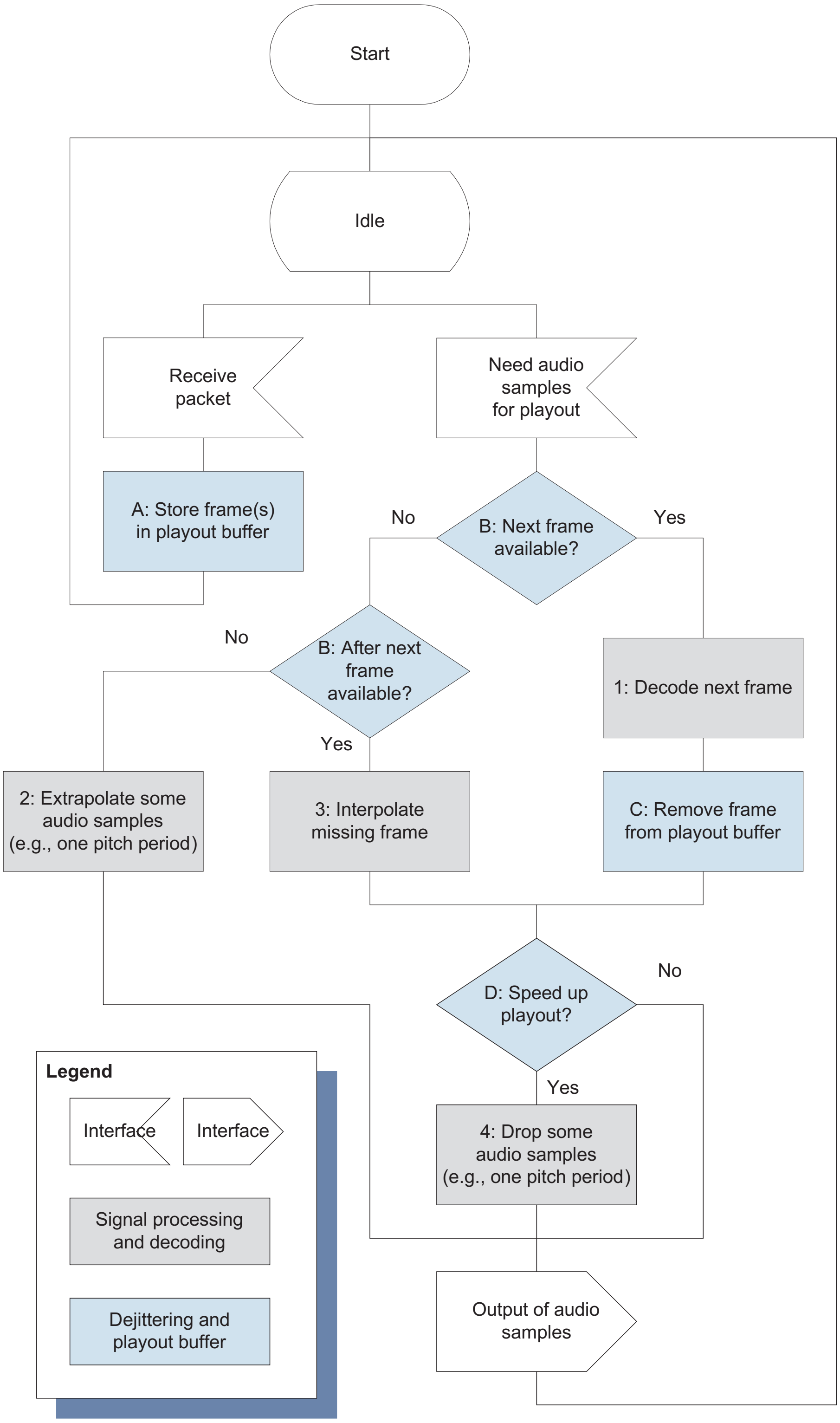

Start
Idle
Receive packet
Need audio samples for playout
A: Store frame(s) in playout buffer
No
B: Next frame available?
Yes
No
B: After next frame available?
Yes
1: Decode next frame
2: Extrapolate some audio samples (e.g., one pitch period)
3: Interpolate missing frame
C: Remove frame from playout buffer
No
D: Speed up playout?
Yes
4: Drop some audio samples (e.g., one pitch period)
Output of audio samples
Legend
Interface
Interface
Signal processing and decoding
Dejittering and playout buffer

Figure 3.2: Design of a common decoding, concealment and dejittering (CDCD) algorithm.

The length of the extrapolated audio signal can be the length of a typical speech frame. However, the CDCD is free to decide to extrapolate for any other duration. For example, it might be reasonable to extrapolate for one or more pitch periods to limit the perceptibility of the concealment.

3. *The next frame is not available but the after next frame has arrived already.* Then, instead of a extrapolation, it might be more useful to conduct an interpolation of the previous and upcoming audio signal. Also, it is a reasonable assumption that the missing frame is not too late but it has been lost because packet reordering typically happens seldom (<0.1%) . Thus, the CDCD can do an interpolation.

- After step 2 or step 3, either
  1. the audio signals are given to the sound card, or
  2. if the dejittering unit decides to speed up the play out of the audio signal, some pitch periods or an audio segment of the concealed signal are dropped (Function 4).
- Now, the CDCD might notice that it did not provided the audio signal to the sound card on time. Then,
  1. it might request the audio sink to let him more time the next time,
  2. it might request the encoder to reduce the complexity of the used algorithm (e.g. lowering the sampling rate),
  3. or it might select an algorithm which can generate a block of audio in less time (e.g., instead of a good concealment a simple block repetition).

At the encoding side the interface between audio source, codec and network control is easier (Figure 3.3). The audio source sends the encoder received blocks of audio samples which the encoder converts into compressed one frame. Usually, the speech frame is transmitted in on RTP packet to the receiver. Many VoIP applications place multiple frames into one RTP packet. However, we assume that the encoder is responsible of producing longer frames because it can take advantage of reducing redundancy which is present within a large frame. For example, in case of SBC the header can be dropped.

The encoder should be controlled by a number of parameters that control the operating point of the encoder similar to the SBC or the proposed SILK codec [32]. All controlled parameters can be changed during regular operation of the codec without interrupting the continuous audio stream from encoder to decoder. As in the SILK codec and in a previous publication [33], we suggest to make the sampling rate, the bit rate, the packet rate, the packet loss resilience, the complexity and the use of DTX. It should be noted that the switching of encoding modes might cause a distortion (for example, the switching of the AMR coding rate causes a click [31]). Thus, the encoding side should try to conceal the audible effects of a mode switch. Typically, this includes also a concealment operation at the receiving side.

3.3 Functional Description of the RORPP PLC

Because SBC does not include packet or time concealment, we decided to implement the CDCD design with a modified loss concealment algorithm that has been defined in ITU G.711 Appendix

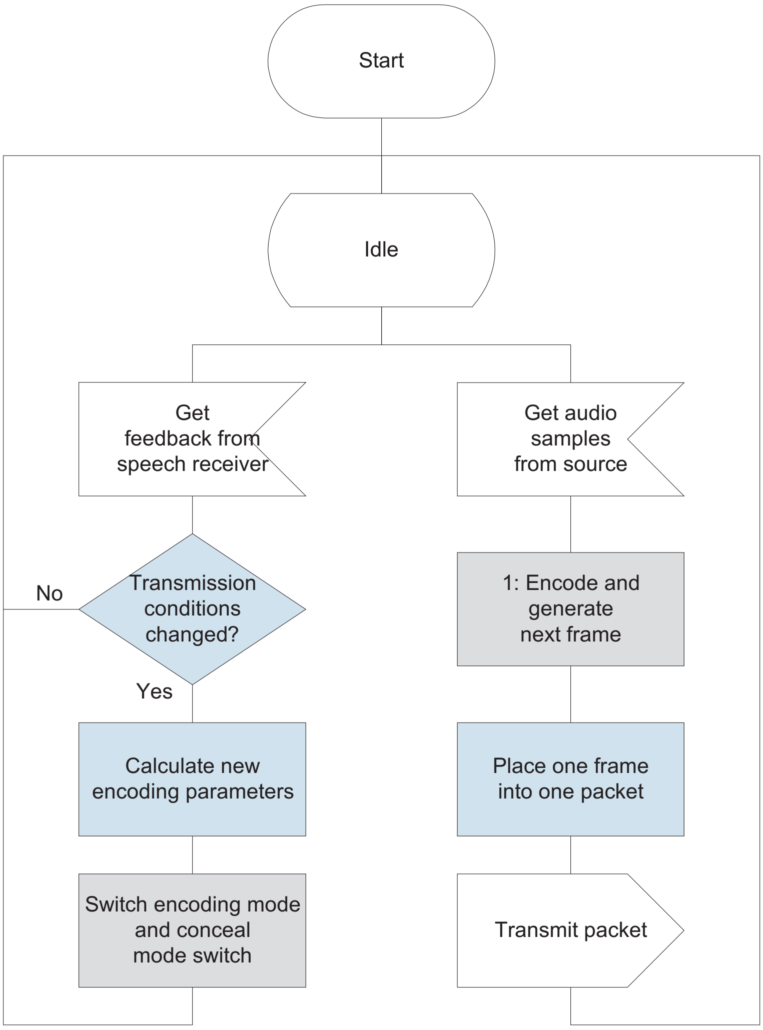

Start
Idle
Get feedback from speech receiver
Get audio samples from source
No
Transmission conditions changed?
Yes
1: Encode and generate next frame
Calculate new encoding parameters
Place one frame into one packet
Switch encoding mode and conceal mode switch
Transmit packet

Figure 3.3: The design of the encoding side.

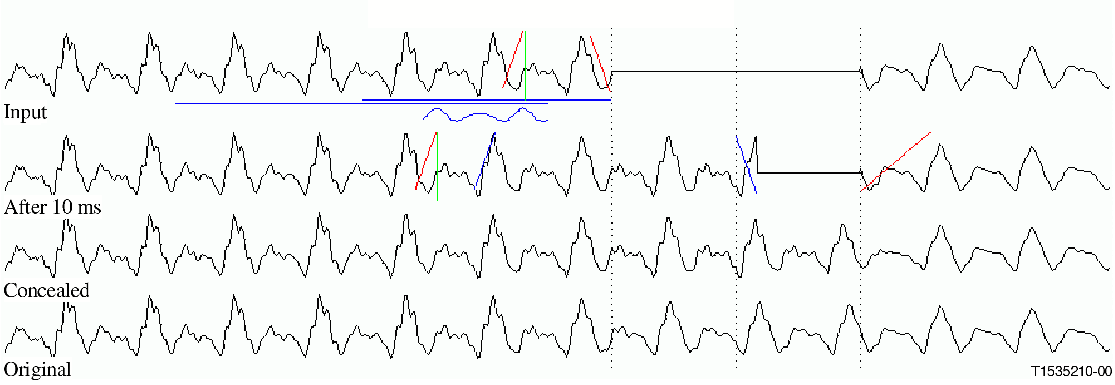


Figure 3.4: Concealment algorithm in G.711 Appendix I [3].

I [3]. Actually, it is “a high quality low-complexity algorithm for packet loss concealment with G.711” and has been designed for narrow band speech transmission and uses a reserve order replicated pitch periods (RORPP) algorithm to replace the missing speech segment. As compared to the numerous other loss concealment algorithms which might perform better in case of frame losses [34], it has a couple of benefits. It has a low complexity, it does not require patent licenses, it is independent of the encoding and decoding algorithm and its source code is open-source via the ITU open-source software license. Thus, we decided to extend it for full-band audio and dual channel operation.

The ITU G.711 Appendix I PLC algorithm works at a sampling rate of 8000 kb/s and has been defined for 10 ms frame consisting of 80 samples. If a speech frame is received successfully, the PLC stores a copy of the decoded output signal in a circular history buffer of a length of 48.75 ms (390 samples). In addition, the speech signal is delayed by 30 samples causing an algorithmic delay of 3.75 ms. This algorithmic delay, used for an Overlap Add (OLA) at the start of an erasure is required for a smooth transition between the real and concealed signal.

In case of a frame loss, the PLC estimates the pitch period [35] of the signal stored in the history buffer. It cross-correlates the last 20 ms of speech with the same speech signal but delayed between 5 ms (40 samples) to 15 ms (120 samples). Typically, pitch periods of frequencies ranging between 66 and 200 Hz can be detected. The cross-correlation is done in two steps. First, the cross-correlation is applied on a speech signal that has been decimated by 2:1 having a sampling rate of 4000 Hz. After the best match has been found, a fine grain search is performed on the original speech signal to find the precise length of the last pitch.

If the first frame is lost, the concealed speech segment is generated by repeating the last 1.25 pitch periods. The loudness of the concealed segment is not changed. To insure a smooth transition between the real and the synthetic signal and between multiple pitch periods, an Overlap Add (OLA) operation is performed using a triangular window of one fourth of the pitch period, both at the start and the end of the lost frame (Figure 3.4). If more than one frame is lost, the synthesized signal contains not only the last pitch period, but also others. Also, the loudness is decreased.

To support SBC, we need to make the following extensions. First, our PLC algorithm works on any arbitrary sampling rate. To keep the complexity low, we conduct the first step of cross-correlation on a downsampled signal of about 4000 Hz. Thus, if the original audio signal has a sampling rate of 48 kHz, it is delimated by 6:1. We support stereo concealment by using two mono channels. Instead of the fixed algorithmic delay of 3.75 ms, any delay up to 3.75 ms can be selected to further reduce the overall transmission delay. Also, the algorithm has to be changed to work on block of an arbitrary number of samples instead of constant sized frames which helps to reduce the transmission delay and this is something which is required because SBC works with frame size of variable size.

One main improvement of our PLC is that it is supporting both frame erases and time concealment. To reduce the algorithmic delay of time concealment algorithms, we combine both loss and time concealment into the same algorithm, which works as follows. In case a speech frame scheduled for playout is not received punctually, the PLC does not know whether the packet will still arrive lately or it has been dropped. Thus, at this point of time, the PLC cannot distinguish whether to use time or loss concealment. Consequently, we start to conceal the gap or the lost packet with the same concealment strategy (namely RORPP). The decision on whether to conceal jitter or loss is done at a later point of time, .e.g. until the current or the next frame is received.

To support time concealment we added the following four functions.

1. The first works on blocks of given size (e.g. the current frame size). If the playout needs to be extended, a block of audio after frame *n* is concealed. Afterwards, frame *n+1* is decoded.

2. To fasten the playout, we added a function which simply skips one frame. For example, frame *n* is played normally, frame *n+1* is skipped and frame *n+2* follows immediately after frame *n*. To reduce the resulting distortion, we mix the beginning of the skipped frame with the following frame.

3. Alternatively, if the skipped frame has not been received, we extrapolate the pervious frame (as for PLC) and mixed the generated signal with the following frame.

4. A further optimization can be made, if one does not work on frame sizes but extends or shrinks the playout time by on pitch period. To determine the pitch period, we use the same cross-correlation function as for the PLC.

Our PLC is based on the source given in ITU G.711A1 but significant numbers of changes were required. We changed the algorithm work on samples not on frames of size 10 ms. The reason behind this change is straight forward. Neither ITU G.711 nor RTP define how large the µ- or a-law decoded frames have to be. Thus, any sender can select any arbitrary number of samples that he places in a RTP packet. In addition, it is not for sure that the acoustic playout works on frame sizes of 10 ms. Thus, to support a packet loss concealment working on 10 ms additional intermediate buffers are required with increase in the algorithm delay. Instead, it is better to support blocks with any arbitrary number of samples.

We also made support variable sampling frequencies, which can be selected at start-up time. Besides the sampling rate, the constructor also defines the lower and higher frequency of the pitch, which is considered for reconstruction of the concealed signal. In case of losses, the PLC overlaps a quarter of the pitch period. The algorithmic delay of the PLC is thus bounded to the lowest pitch supported pitch frequency. Thus, if the pitch frequency is 66.6 Hz, the algorithmic delay is $\frac{1}{4 \bullet 66.6 Hz} = 3,75 ms$. Using the constructor, one can thus also control the algorithmic delay and tweak it if ultra-low delay is required.

In order to test the quality of the loss concealment, we added our implementation of the PLC into the open-source VoIP client Ekiga. Ekiga is one of the most common VoIP client solutions running under Linux. Ekiga G.711 did not include a packet loss concealment before.

4 Human and Objective Audio Assessments

In both the SBC and PLC algorithms, a couple of different parameters are variable. These parameters have different effects on quality, rate and delay. Here, it is interesting to address the question which of the parameter setups providess an optimal trade-off at a particular operating mode. We can calculate algorithmic delay and the bit and frame rates easily but it is not known how the coding parameters influence the audio quality.

ITU Recommendation BS.1116 describes a procedure on how to judge the impact of small audio degradations caused by transmission systems. In listening tests those degraded audio samples are rated relative to a reference signal. Typically, a scale called subjective difference grade (SDG) consisting of the values 0 (Imperceptible), -1 (Perceptible but not annoying), -2 (Slightly annoying), -3 (Annoying), and -4 (Very annoying) is used. These tests have to be done repeatedly multiple listeners and the results are then averaged. In order to obtain statistically significant and repeatable results, these tests have to be done under well controlled experimental conditions with sufficient number of experienced listeners.

For intermediate audio qualities, the MUSHRA method as specified in ITU-R BS.1534-1 [36], is more suitable. Because of the expect kind of distortions, we chose MUSHRA to judge the audio quality of SBC and PLC.

We aim to assess all coding modes of SBC and a large range of parameter settings of PLC for multiple audio samples having largely different contents. A large number of tests are required to find potential programming mistakes. However, then the number of subjective tests are exorbitantly high and we are not able to complete them in reasonable time scale. Thus, we use objective assessment method to judge all parameter combinations and subjective measurements to judge a part of possible sample items. Because objective audio quality evaluation is not as good as the human interrogation, we need to compare the results of subjective and objective assessment to figure out the precision, weaknesses and strength of the objective assessment algorithms.

4.1 MUSHRA Tests

The signal items used for our MUSHRA tests are based on the audio items given in ITU-R BS.1387 and the „Kiel Corpus Vol. 1“. We generated anchors consisting of IRS48 filter for narrow-band, P341 filter for wideband, a super-wideband filtering at 14 kHz (all made with the ITU-T G.191 software), and a version sampled at 8000 Hz version of the samples. It should be noted that the samples of the Kiel Corpus had a sampling rate of 16000 Hz and were up-scaled to 48000 Hz if required.

We conduct two rounds of listening-tests, in each we interrogated 6 subjects. In the first round, we used sample items generated with the PLC algorithmic, which had random loss frame losses at a rate of 2 and 8%, a sampling frequency of 8000 or 48000 Hz, and a frame size of 2.5 ms and 10 ms. Also, we added anchors to the tests. We used the samples called “hpte005.16”, “hpte011.16”, “kkoe025.16”, “kkoe026.16”, “rtde031.16”, “rtde045.16”, “ugae063.16”, and “ugae078.16”, which contains sentences spoken in German taken from the Kiel Corpus, and from ITU BS.1534 “refpia01” (Piano), “refsfe01” (English female), “refsme01” (English male), “refsmg01” (German male), and “refveg02” (Suzanne Vega singing). In the second round of listening tests, we used the early version of the Linux SBC implementation to generate samples with 4 and 8 subbands, in the LOUDNESS allocation mode, with 16000, 32000 or 48000 Hz sampling rate and a bitpool size between 10 and 40.

User experiments on assessing the audio system are performed with normal hearing and paid subjects. We have used Sennheiser HD 280 PRO headphones and the MATLAB software “MUSHRAM 1.0“ by E. Vincent. In total, we got 584 assessment values, each ranging from 0 to 100[1].

4.2 Comparison between Subjective and Objective Audio Quality Tests

Since about 20 years, researchers developed computation method for perceptually assessing the quality of audio transmission. In 1994, the ITU tried to standardized an objective audio assessment methods but all seven proposed algorithms did not fulfilled the requirements given afford hand. Thus, the ITU developed jointly an improved algorithm that was called perceptual evaluation of audio quality (PEAQ) [38]. It was published in the document ITU BS.1387 in 1998 [10]. PEAQ is intended to predict the quality rating of low-bit-rate coded audio signal. Two different versions of PEAQ are provided: a basic version with lower computational complexity and an advanced version with higher computational complexity.

Beside the MUSHRA values got from the listening-only tests, we also applied ITU BS.1387-1 (PEAQ) for an assessment of audio quality and ITU P.862 (PESQ) for the evaluation of speech quality using the sample items as in the listening-only tests.

PEAQ supports two modes: the basic version (BV) for a fast and low complexity assessment and the advanced version (AV) for a better but slower assessment. We used PESQ for narrow and wide band assessment of the down-sampled but not IRS filtered sample items. In addition, we used a recently published algorithm by Creusere et al. [12], which uses parameters similar to those that are calculated by the basic version of the BS.1387 algorithm and combines them with an energy equalization truncation (EET) threshold to calculate MUSHRA estimate.

Throughout this publication we will use the raw calculation results of PEAQ denoted as Objective Difference Grade (ODB-BV and ODG-AV respectively), the raw results of PESQ described as PESQMOS-NB and PESQMOS-WB and Creusere’s values denoted as MUSHRA-LQO and EET.

In the following, we assume that the MUSHRA method and the six objective assessments have an interrelation. The questions that we like to answer are the following. How does the interrelation look like? How precisely are the objective assessments as compared to the human references? Which kind of distortions do the objective methods judge more precisely and for

[1] Partly, these results have been presented already in [37].

which kind of distortions do they fail?

Six scattered plots are displayed in Figure 4.1, which are calculated using the individual (not averaged) MUSHRA test values and their corresponding objects results. The smoothed line in the figures has been calculated by using a local fitting algorithm and shows the relation between subjective MUSHRA and the objective results. More precisely, the plots show a roughly linear relation between MUSHRA values and ODG respective PESQMOS values for a MUSHRA value range between 40 and 100. For the range of 0 to 40 the lines in the ODG-AV, PESQMOS-NB, and PESQMOS-WB figures are indifferent. The EET threshold decreases with increasing audio quality and MUSHRA-LQO and MUSHRA-LQS show a linear correlation for a range between 60 to 90 and 0 to 40 respectively. The MUSHRA-LQO estimate seems not to be available to distinguish amount better audio qualities.

Overall, in Figure 4.1 the measurement “dots” are scattered widely and the plots have many outlying measurement results. The mean residual errors between the lines and the measurements results are quite large. Thus, these scatter plots can only be considered as a first step in understanding the relation between subjective and objective ratings.

4.3 Identifying the Outliers

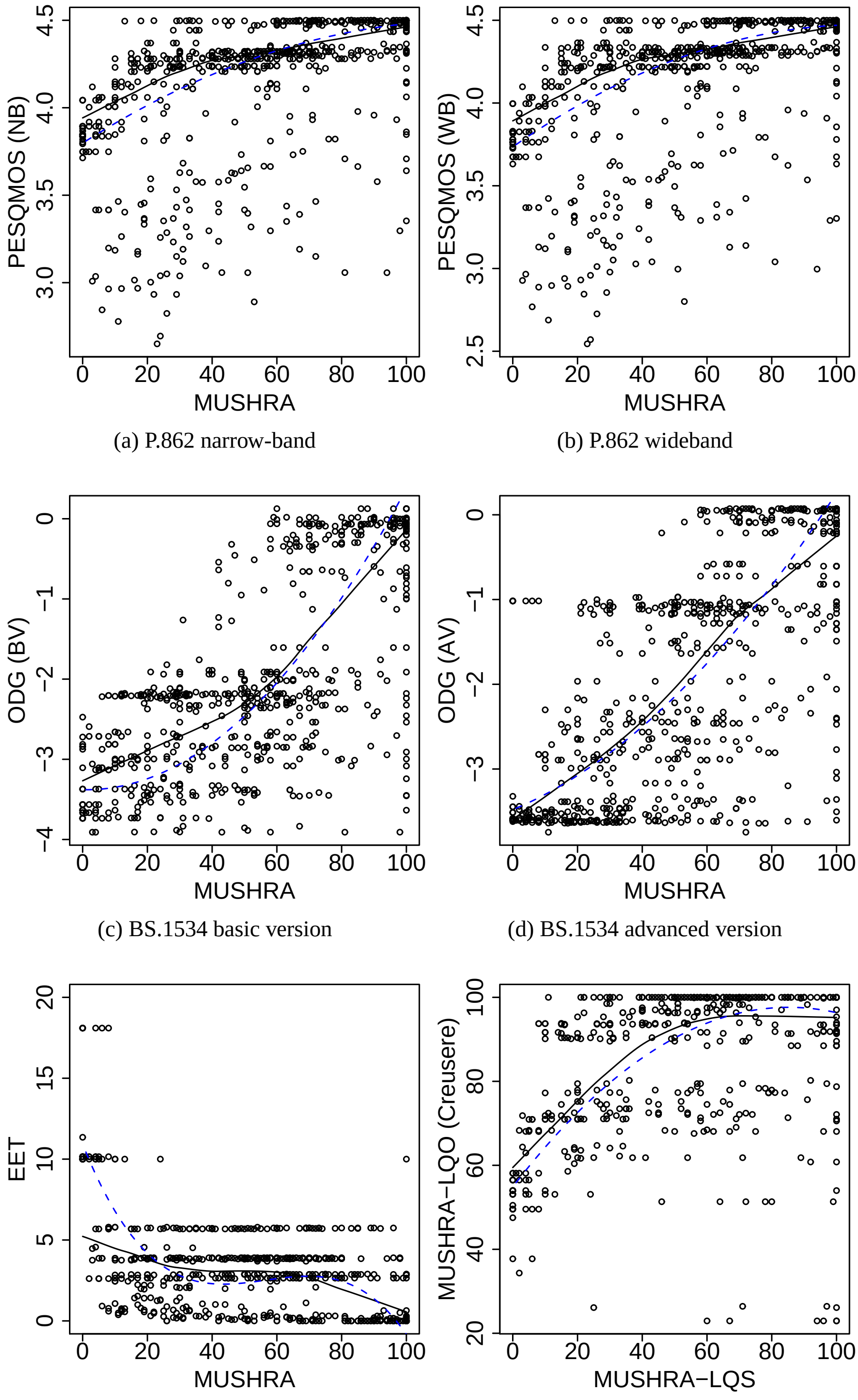
The MUSHRA testing procedure suggests to a post-screening of subjects to figure out whether any subject results show any inconsistencies with the mean result. Indeed, we identified some results where one subject was unable to judge the audio quality as precise as the others. Therefore, we removed his results. In another case, one of the subject did not change the ratings of a couple of samples, may be due to some technical reasons. Therefore we removed all his ratings which had an unchanged rating of 100.

Also, the objective assessment results seemed to be obvious outliers. For example, The MUSHRA-LQO value could not be calculated for the sample from the Kiel Corpus. This may be due to a programming error in the public available MATLAB implementation in combination with the lack of higher frequencies in the Kiel Corpus. Also, some results must have been removed due to other technical errors. In addition, some EET were negative—amusingly a miscalculation. We removed them and also the corresponding MUSHRA-LQO values that were calculated using the erroneous EET values.

We identified many more outliers caused by inattentive human ratings or failures in the objective assessment algorithms. However, removing them would have falsified the results which are based on the occurrence of the voted results. Thus, we left them in the data sets.

4.4 Quality of Human MUSHRA ratings

Humans do rate the quality of audio samples differently. Partly, this is due to different taste and a different interpretation of the rating scale. For example, we identified one person who was unable to rate the audio quality reliably. At this point a few questions arise, how well do the other individuals rate the audio quality? What is the difference between individual results and the average MUSHRA ratings? In Figure 4.2, we compare two individual MUSHRA ratings with the average MUSHRA voting and calculate a linear regression between both sets of data.



(a) P.862 narrow-band

(b) P.862 wideband

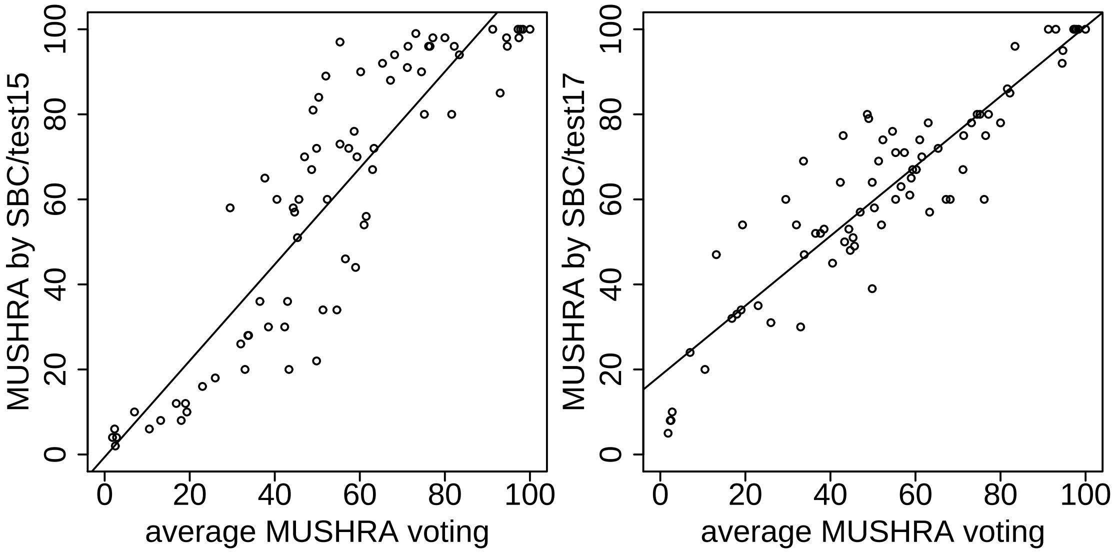
(c) BS.1534 basic version

(d) BS.1534 advanced version

(e) Creusere's EET parameter

(f) Creusere's MUSHRA prediction

Figure 4.1: Comparison between subjective values of the MUSHRA tests and objective assessments (dots) and a smoothed curved calculated by a local polynomial regression fitting. The dotted lines are based on a polynomial function (refer to Section 4.5).



(a) LR with $R^2 = 0.8$, $RSE = 14.9$ and $y = 1.13x - 0.593$.

(b) LR with $R^2 = 0.835$, $RSE = 9.64$ and $y = 0.821x + 18.6$.

Figure 4.2: Comparison between individual MUSHRA ratings and the average voting results displaying the linear regressions.

Figure 4.2a shows that the audio quality is more extreme than the averaged rating, it appears that more samples at high quality and lower quality has been rated than the average.

Figure 4.2b displays a rating behaviour in which the MUSHRA ratings were higher than the norm. However, this time the ratings are well distributed over the entire scale. Both ratings have a correlation coefficiency of $R = 0.89$ and $R = 0.914$ respectively.

4.5 Quality of the Objective Assessments

Lines in the Figure 4.1 show a first mapping between subjective and objective results. In the following section we try to enhance this mapping function. We bring it in closed form to have a look at conditions in which the objective assessment algorithms perform better and show a better correlation to the subjective results.

Instead of the smoothed splines shown in Figure 4.1, which map subjective to objective results, we calculate a linear (first order), quadratic (second order) and cubic (third order) regression to match the averaged MUSHRA ratings with the objective results. The MUSHRA ratings were weighted to consider the fact that the averaged MUSHRA ratings are based on varying number of individual ratings. As metrics of the fitness we use the Residual Square Error (RSE) and the coefficient of determination (R^2) value, which ranges between 0 (none relation) to 1 (perfect match). In case of a linear regression, the coefficient of determination is the square of the sample correlation.

Figure 4.3 displays the scatter plots and regression analysis displaying objective vs. subjective

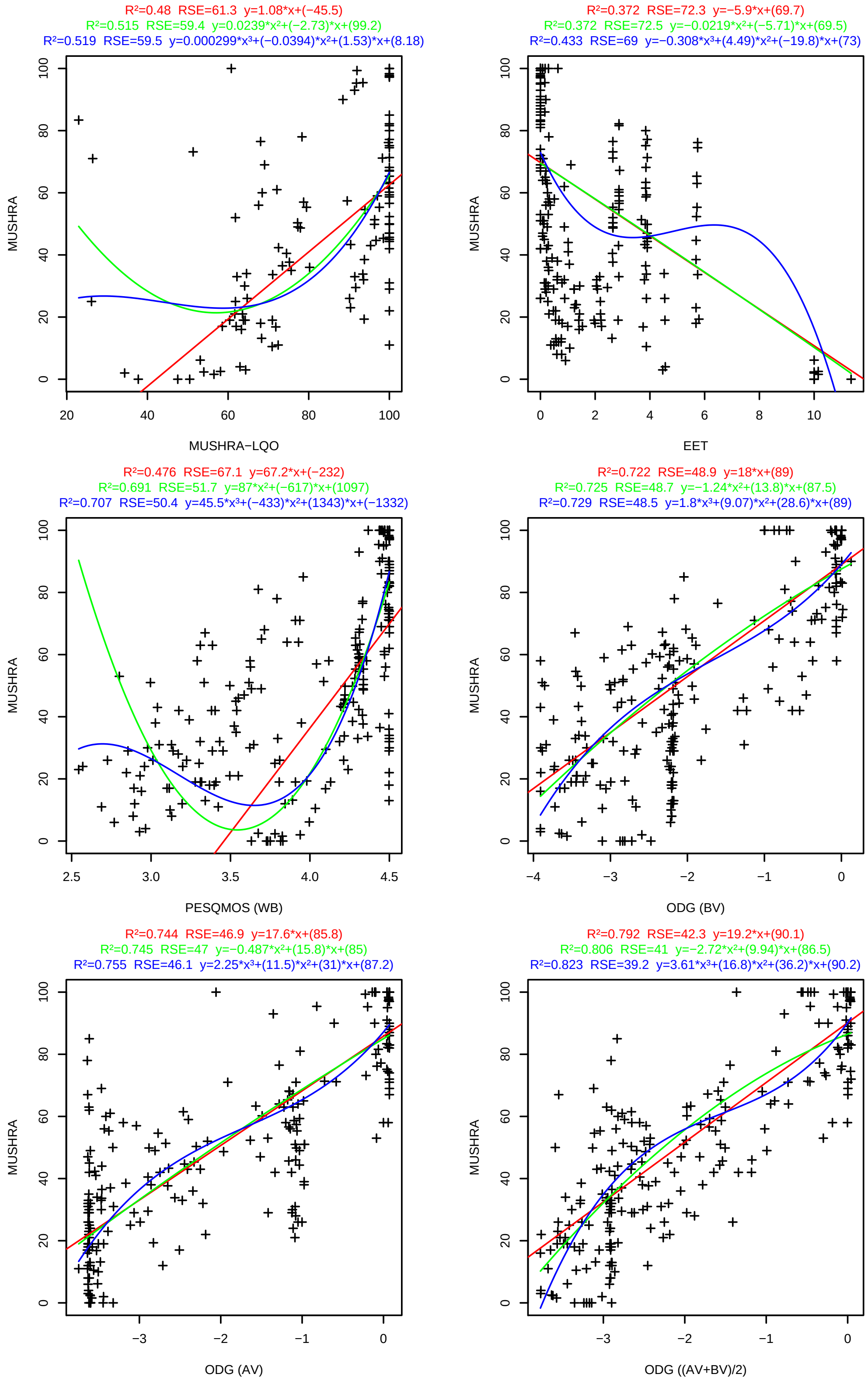


Figure 4.3: Regression analyses showing the relation between subjective and objective results

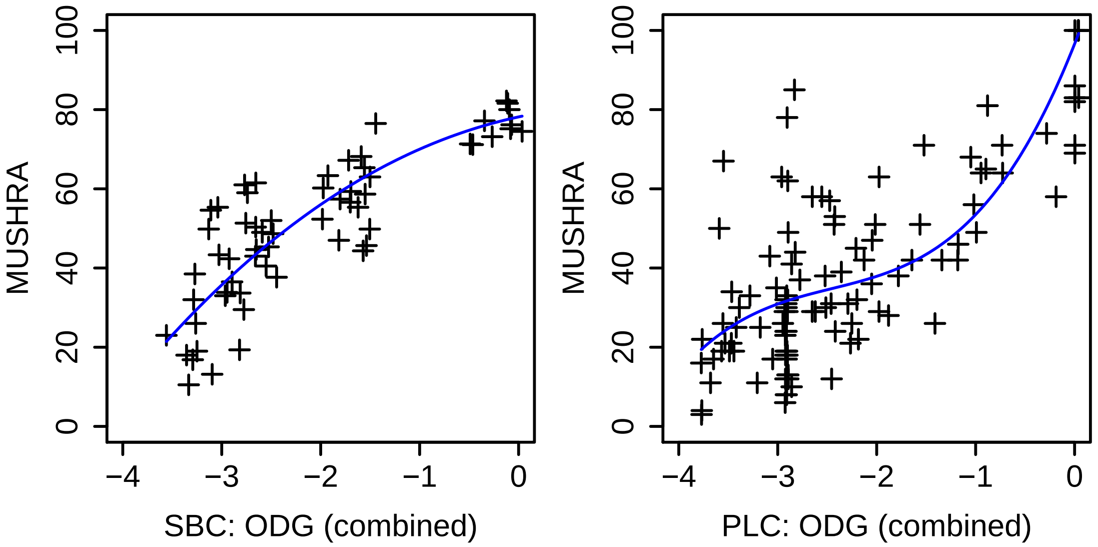


Figure 4.4: Regression analyses showing the relation between subjective and objective results for sampled items degraded by only PLC or SBC.

ratings. The plot of PESQMOS (WB) vs. MUSHRA shows that PESQ cannot be used to predict MUSHRA results. The estimated mapping function does not even increase statically. This is an expected result because PESQ has been designed for speech not for audio samples.

Also, both Creusere's MUSHRA-LQO and EET show a bad prediction performance. Quite many values are clearly outliers. Further analyses have shown that the MATLAB implementation does not correctly implement the PEAQ algorithm. Further debugging of Creusere's implementation of EET and PEAQ is likely to lead to better results.

Better results are achieved with the basic version of PEAQ showing a measure of fitness of $R^2 = 0.854^2$, if a cubic regression is used. The advanced version of PEAQ performs slightly better with $R^2 = 0.869^2$. Both are not very good in judging the quality of very good or very bad samples. Again, this matches the usage descriptions of PEAQ, which is described as intended for intermediate quality.

Interestingly, if one combines both the results of the basic and advance versions of PEAQ, the prediction performance even increase. For example, averaging both ODG values and mapping them to MUSHRA yield a goodness of fitness of $R^2 = 0.907^2$, which is the best prediction of audio quality.

In Figure 4.4 we compare objective and subjective ratings only for a selection of samples. We have done the mapping between all SBC encoded samples. The coefficient of determination in case of the third order mapping is $R^2 = 0.869^2$. Similar, in the case of PLC, the mapping is $R^2 = 0.801^2$.

Still, if one compares the objective results with those of a individual rating, the performance of PEAQ does not outperform the subjective ratings of a single human. As such, the quality of PEAQ cannot be considered better than informal listening tests.

In Figure 4.5 we compare objective and subjective ratings only for a selection of samples.

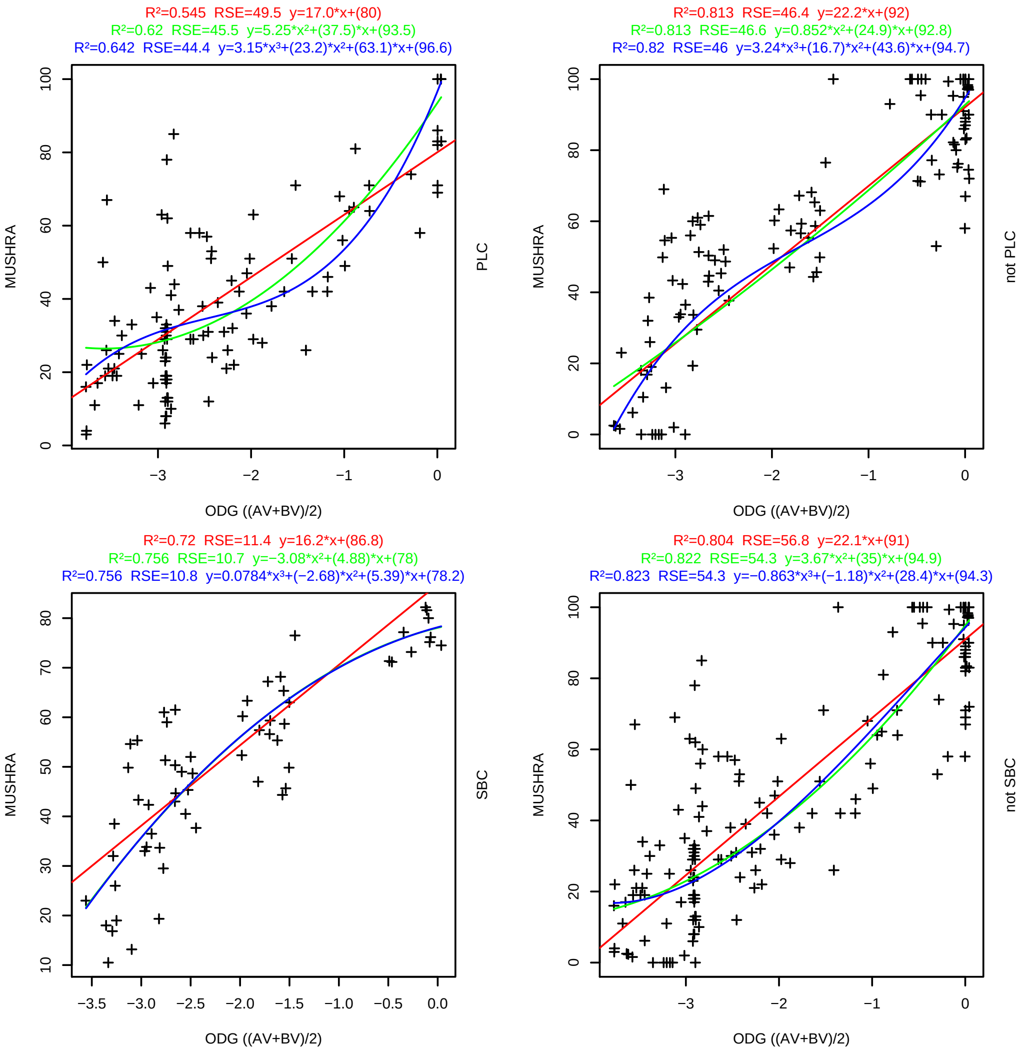


Figure 4.5: Regression analyses showing the relation between subjective and objective results for sampled items degraded by PLC or SBC.

We have done the mapping between all SBC encoded samples and for all samples not encoded with SBC. The coefficient of determination in case of the third order mapping is $R^2 = 0.869^2$ respective $R^2 = 0.907^2$. Similar, in the case of PLC, the mapping is $R^2 = 0.801^2$ for sample items with packet losses and $R^2 = 0.905^2$ for those without.

Several studies compared the PEAQ's audio prediction to those gained from subjective tests. Treurniet and Souldore [39] studies the correlation of eight audio items used for 17 different low-rate audio codec settings and the rating of 17 expert listeners. The correlation between those ratings where $R = 0.85$. The correlation was be enhanced significant to $R = 0.95$ if the averaged rating results over different audio content was taken.

Huber and Kollmeiner developed an enhanced perceptual audio assessment that is based on psychoacoustically validated, quantitative model of the human auditory processing [40]. The authors compare their algorithm called PEMO-Q with PEAQ. For a known data set, PEMO-Q show a linear correlation of $R = 0.9$ to subjective ratings. PEAQ-basic and PEAQ-advance show under the same conditions correlations of $R = 0.89$ and $R = 0.87$. Better correlations were achieved if the test set were restricted to curtain type of content or distortions.

Grancharov and Taleb tested on how well PEAQ can measure the quality of different implementations of G.722.1 FB. Correlation coefficiency ranged between 0.86 and 0.97 after third order monotonic polynomial mapping of subjective and objective ratings [41].

Voldhaug et al. have used PEAQ to judge the distortions caused by packet loss concealment in [42]. They found that it perform as worse at $R = 0.57$ because a number of outliers. If the results of multiple loss conditions and sample items are averaged, then the authors found the cross correlation at $R = 0.84$. Also, the PEAQ rated the quality due to loss impairments less badly than humans.

5 Evaluation

5.1 Bluetooth SBC: Quality vs. Bit rate

As described in Section 2, SBC allows to paramete rize this operation to a wide range. Even though, the A2DP defines some recommend parameters to use, we are interested in understanding, which parameter sets are best at a given bandwidth.

To address these questions, we run extensive simulations with PEAQ varying both the parameters and the reference samples. For all the reference samples files “refcas” (castanets), “refcla” (clarinet), “refclv” (Calves), “refflu” (flute), “refglo” (glockenspiel), “refhrp” (harpsichord), “refpia01” (piano), “refryc” (jazz music by Ry Cooder), “refsax” (saxophone), “refsb1” (bag pipe), “refsna” (snare drums), “refsop01” (opera), “reftpt” (trumpet), “reftri” (triangle), “reftub” (tuba), “refveg01” (Suzanne Vega singing), “refveg02” (ditto), and “refxyl” (xylophone) in stereo modes and mono versions of the afore mentioned samples plus “refsfe01” (English female), “refsfe02” (ditto), “refsme01” (English male), “refsme02” (ditto), “refsmg01” (German male), “reftam” (tambourine) , we calculated all coding modes varying the allocation mode (SNR, LOUDNESS), the number of subbands (4 and 8), the number of blocks (4, 8, 12, 16), the coding mode (mono, stereo, joint stereo) and the bit pool value (10, 12, 14, 18, 19, 25, 29, 31, 40, and 50). Overall, 4800 PEAQ ODG-BV and ODG-AV have been calculated. In the one channel mode, we have compared the degraded files to the mono version of the references file. In the stereo modes, the degraded samples were compared with the original stereo reference file. In addition, we approximated the quality for remaining bitpool parameters between 11 and 49 with a natural spline function in order to save time.

In Figure 5.1, we plot the averaged ODG values versus the coding rate. The ODG results of parameter sets, which differ only in the bitpool values, are interconnected by lines. We also highlighted the best parameter sets with coloured lines. In the mono mode up to a rate of about 96 kbps, the 16 kHz, 16 blocks, LOUDNESS coding mode is the best. Then, between 96 and 72 kbps, the 32 kHz sampling rate should be chosen. Further up the axis multiple best coding alternate at fast pace.

In the stereo mode, choosing the right mode is simpler. Up to 106 kbps, the 16 kHz, 16 block, LOUDNESS mode is best. Both the stereo and joint-stereo mode seems to encode equally good. Then, up to 237 kbps, the 32 kHz sampling rate is the best. At higher quality, the 44.1 kHz stereo encoding mode can be chosen.

Some of coloured lines match those recommend in the A2DP standard (and Table 2.2). However, if a lower audio quality is required, the results suggest to use the 32 kHz coding mode. Also, the joint stereo mode does not increase significantly the audio quality as compared to the stereo mode.

In Figure 5.2, we display the best mode when both bit rate and delay are constraint. Most of the time, up to a rate of 100 kbps, the 16 kHz sampling rate is best, till about 200 kbps, the

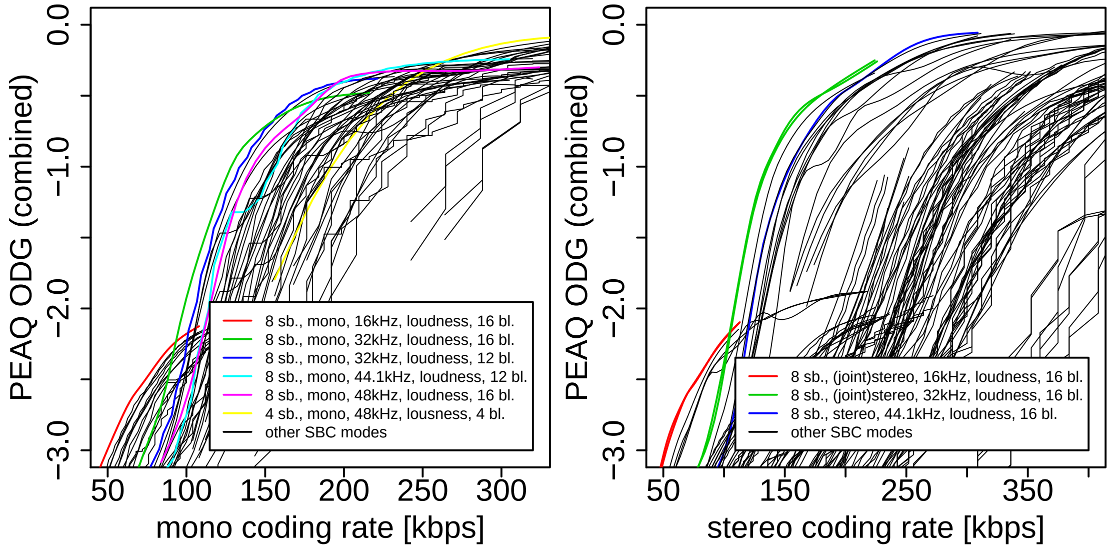


Figure 5.1: Using SBC for mono and stereo audio

32 kHz sampling rate, till about 270 kbps, the 44.1 kHz sampling rate, and above 270 kps, the 48 kHz sampling rate. Below a delay of 4 ms and above 260 kbps, the 4 subbands mode is preferable. The 8 subbands mode is good if used between 100 and 260 kbps for delay of 4 ms and above. Based on the results, the LOUDNESS mode performs better than SNR most of the time. Interestingly, the 16 block mode is not always the best but the 12 and 8 block mode can be good choices, too. The 4 blocks mode is only beneficial if used a delay below 1.5 ms.

Similar results are valid for the stereo modes and can be derived from the results shown in Figure 5.2), too.

5.2 Bluetooth SBC: Quality vs. Gross rate

In packetized networks, speech frames are also transmitted in packets, which have packet headers. In the Internet, the size of packet headers can vary depending on the kind of protocol used and whether header compression is applied. In a typical scenario, one frame is transmitted with the RTP, UDP, IPv4 and IEEE 802.3 protocols and thus each packet contains packet headers having 12 bytes, 8 bytes, 20 bytes and 18 bytes respectively. In the end, the gross rate, as measured on the physical layer is much larger than the actual coding rate. Thus, we also consider this gross rate in addition to the coding rate. The gross rate calculates as

$$r_{gross} = r_{coding} + packetoverhead * framerate \tag{5.1}$$

where coding rate give the coding rate of the SBC codec, *packetoverhead* is the number of bits for protocol headers in each packet (typically 58*8=464), and the *framerate* is the number of packets/frames per second.

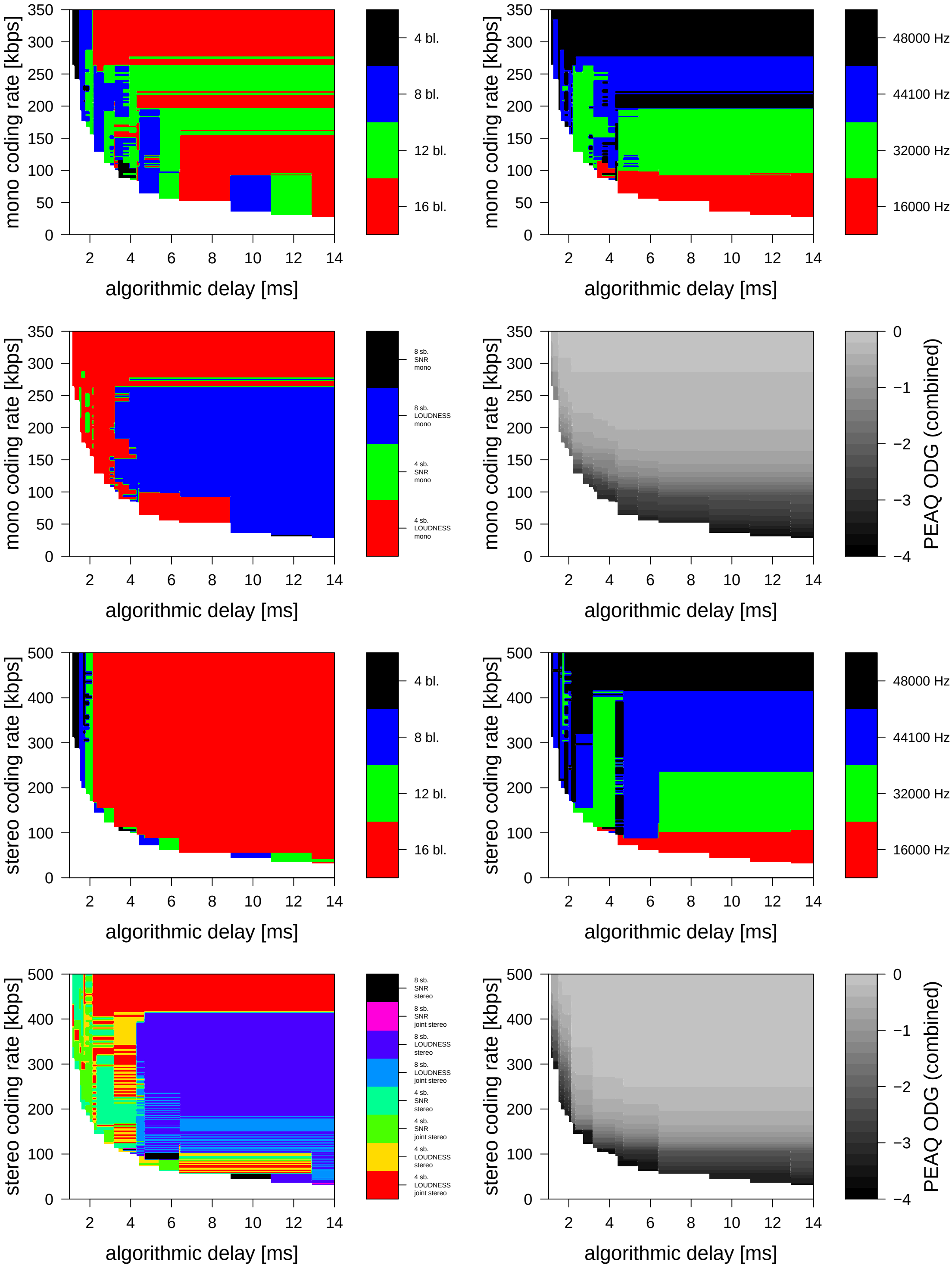


Figure 5.2: Using SBC for mono and stereo audio with rate and delay constrains

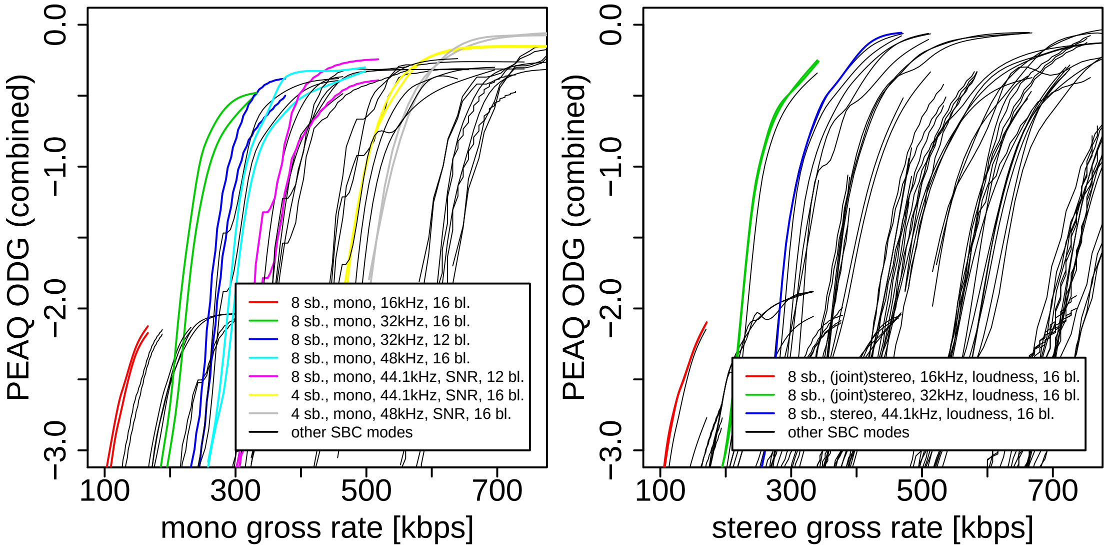


Figure 5.3: Using SBC for mono and stereo audio measuring the gross rate (including RTP, UDP, IPv4, and Ethernet packet headers)

Considering the gross rate, the best coding mode for bandwidth constraint link is shown in Figure 5.3. As compared to the Figure 5.1, the best coding mode hardly changed. However, considering both delay and gross rate (Figure 5.4 and 5.4) difference can be seen clearly. Especially, the 16 block mode is much more important.

5.3 Narrow and Wideband Speech

The Bluetooth SIG standardization group currently considers to use SBC for wideband headsets to transmit the microphone signal. In Figure 5.5 we display the mean ITU P.862 wideband MOS results for speech samples including the Kiel corpus samples and the ITU BS.1387 speech samples (English male, English female, German male, Suzanne Vega singing). For all SBC coding modes, the mode with 8 subbands, 16 kHz sampling rate, kHz sampling rate, LOUDNESS allocation mode, 16 blocks and mono provides the best speech quality. It performs slightly better than ITU G.722 and 48, 56, and 64 kbps.

In addition, we show the results of the SBC 16 kHz coding mode with samples that were shifted by on octave up. We refer to this mode as SBC 8 kHz sampling mode, which however, is not standardized. The measured PESQ values of this mode are even better than of the 16 kHz sampling mode.

5.4 Packet loss concealment

The following results should be objective quality indications of the G.711A1 like packet loss concealment (using a PCM coding). We simulated randomly distributed, single packet losses at

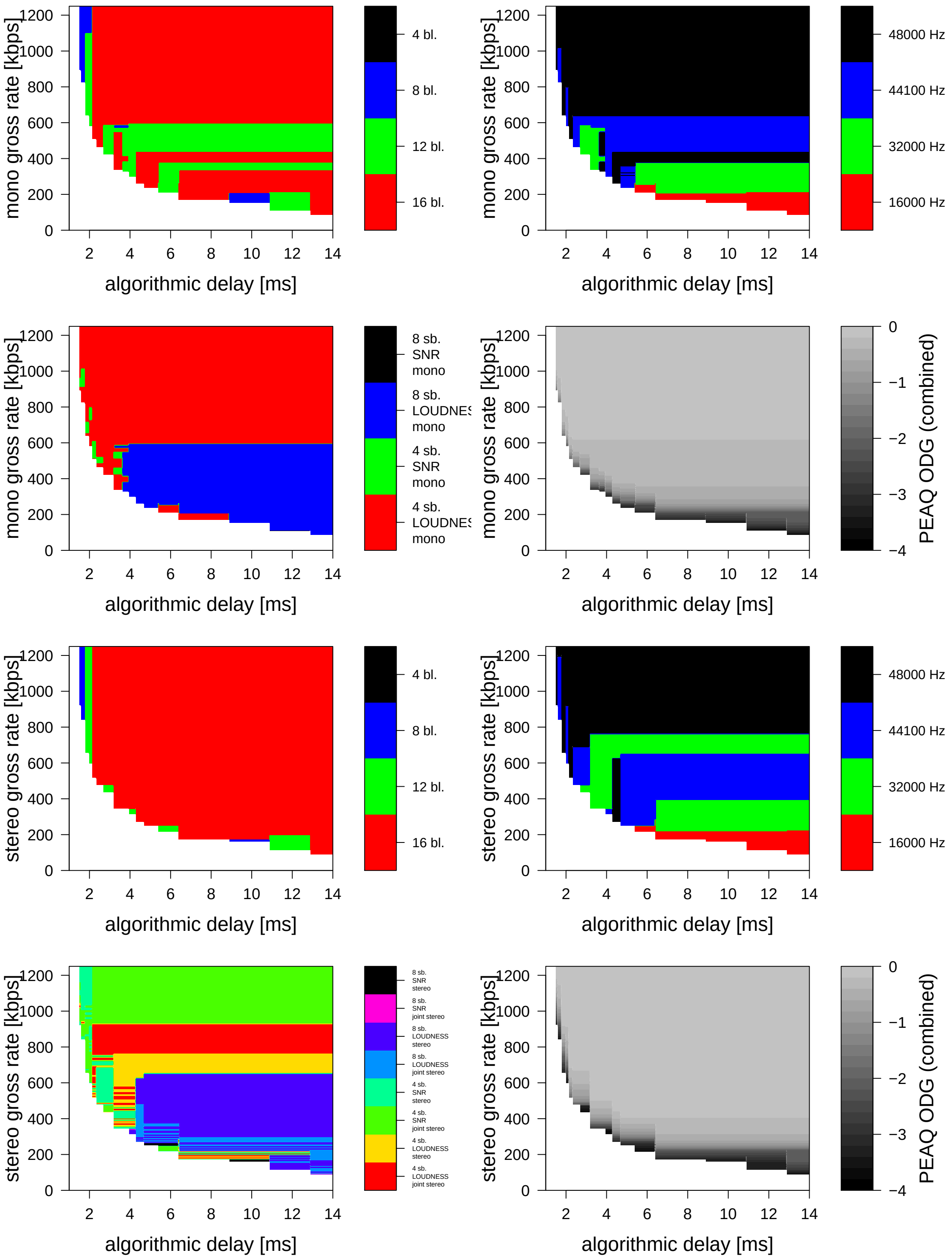


Figure 5.4: Using SBC for mono and stereo audio with rate and delay constrains

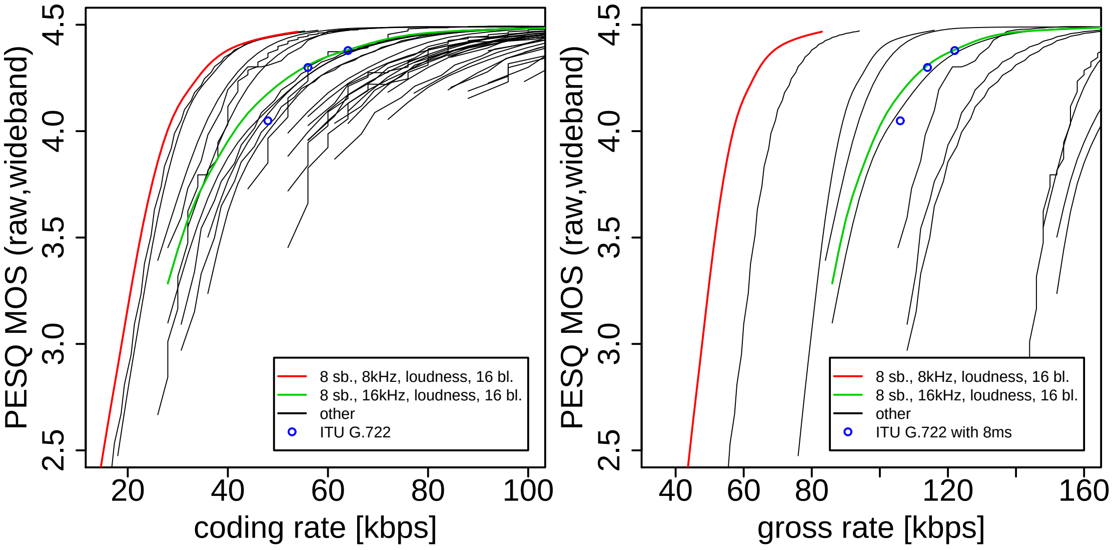


Figure 5.5: Using SBC for wideband speech

loss rates between 0 and 5%. We vary packet size between 2.5 and 20 ms. Figure 5.6 shows the audio and WB speech quality version versus the loss rate. Interestingly, the results show a larger degradation for smaller frames. Only the 20 ms packet size at 32 and 48 kHz, if measured with PEAQ, performs worse than the 10 ms frame size. However, one should note that PEAQ has not been optimized to judge the quality of frame losses.

Next, we varied the algorithmic delay of the PLC between 0 and 3.75 ms. The algorithmic delay PLC comes from the overlap and add period which is required to smooth the transition from non loss audio segments to concealed segments to original audio again. The results in Figure 5.7 show that the higher delay, the better. In any case, an algorithmic delay of 0 (thus no overlap/add) should be avoided.

5.5 Content

Both the SBC and the PLC algorithm might not perform equally well for all kind of acoustic content. To avoid a content specific judgement in the previous tests, we have taken the objective ratings averaged over multiple, different sample files (refer to Section 5.1). This time, we take the average of all the sampling modes but keep the sample file fixed. The results are given in Table 5.1). The 16 kHz sampled speech and noisy instrument such as the snare drum can be compressed rather well. On the other side, single instruments having high tonal sounds such as the glockenspiel, the tambourine, the flute, the triangle and the clarinet are encoded relatively bad. If looking on the measured speech qualities, it is interestingly to note that high, female voices are encoded worse and low male ones. Music such as the opera, the piano and full band speech show an average compression efficiency. Assuming, these are also the most common content, which is transmitted via a hifi-phone.

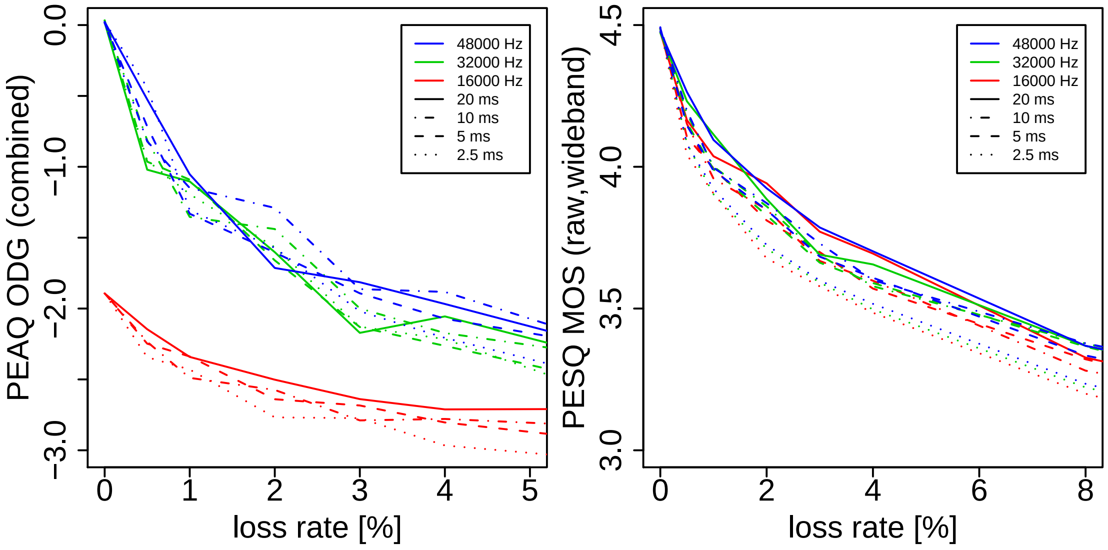


Figure 5.6: Performance of the PLC for different coding rates and frame sizes

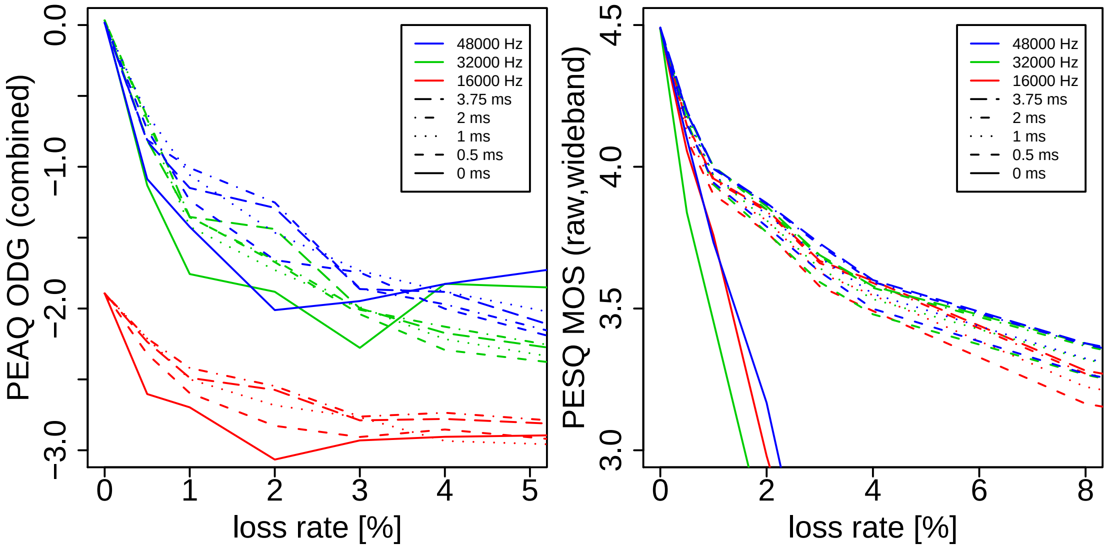


Figure 5.7: Performance of the PLC for different coding rates and algorithmic delays

Sample	*ODG avg.*	*ODG std.*	*Sample*	*PESQ-WB avg.*	*PESQ-WB std.*
refglo	-2.77	1.43	reftam	3.09	1.229
reftam	-2.59	1.43	reftri	3.13	0.783
refflu	-2.39	1.48	refglo	3.54	0.761
reftri	-2.35	1.43	refsax	3.61	0.355
refcla	-2.29	1.42	reftpt	3.95	0.558
refsb1	-2.19	1.52	refcla	3.98	0.762
refryc	-2.13	1.51	refryc	4.06	0.409
refcas	-2.09	1.52	refhrp	4.07	0.532
refsax	-2.07	1.37	refflu	4.09	0.563
reftpt	-2.05	1.47	refsb1	4.12	0.532
refveg02	-2.01	1.55	refsfe02	4.14	0.496
refhrp	-2.01	1.49	refveg01	4.15	0.476
refveg01	-1.99	1.55	*average*	4.16	0.304
refsop01	-1.92	1.41	refxyl	4.16	0.292
reftub	-1.92	1.36	refsfe01	4.17	0.480
refsmg01	-1.87	1.49	refsmg01	4.20	0.335
refsfe01	-1.86	1.55	refveg02	4.20	0.445
refsfe02	-1.85	1.54	refcas	4.21	0.300
refsme01	-1.80	1.52	refpia01	4.24	0.283
refsme02	-1.80	1.53	refsop01	4.27	0.459
average	-1.74	2.07	rtde045	4.29	0.417
refpia01	-1.71	1.30	reftub	4.29	0.400
refxyl	-1.61	1.43	rtde031	4.29	0.408
refclv	-1.47	1.34	rtde040	4.29	0.411
ugae060	-1.47	1.31	kkoe019	4.31	0.346
hpte005	-1.47	1.29	rtde035	4.32	0.379
rtde040	-1.46	1.32	kkoe023	4.32	0.320
ugae078	-1.45	1.29	kkoe025	4.32	0.334
ugae063	-1.43	1.31	hpte004	4.33	0.364
rtde031	-1.43	1.34	hpte011	4.33	0.356
rtde045	-1.43	1.31	kkoe026	4.33	0.330
hpte004	-1.41	1.26	hpte015	4.33	0.368
hpte015	-1.40	1.27	refsme01	4.33	0.311
rtde035	-1.38	1.31	hpte005	4.33	0.329
ugae051	-1.38	1.28	refclv	4.33	0.218
kkoe025	-1.37	1.23	ugae068	4.34	0.336
hpte011	-1.36	1.24	refsme02	4.35	0.293
ugae068	-1.36	1.26	ugae078	4.35	0.316
kkoe023	-1.30	1.22	ugae051	4.35	0.328
kkoe019	-1.30	1.22	ugae060	4.35	0.326
kkoe026	-1.18	1.20	ugae063	4.36	0.338
refsna	-1.16	1.12	refsna	4.43	0.131

Table 5.1: SBC coding quality depending on sample content averaging over all sampling modes. The left side shows the quality measured with averaged PEAQ approach. The right side gives the quality measured with PESQ-WB.

Similar studies for the PLC algorithm have been made. Table 5.2 display the results. If measuring on the ODG scale, it is interesting to observe that full band speech and complex music cannot be concealed well. However, frame losses in wide band speech and single instruments are concealed rather well. The measurements will PESQ-WB show that the concealment performance is low for high (female) sounds.

5.6 Related Codecs

Several encoding schemes can compress an audio signal with very low algorithmic delay. One of the simplest encoding techniques is to use a PCM coding at different sampling rates. Also, a logarithmic quantization of the samples [43] can be considered. The classic logarithmic quantization called µ-Law and A-Law has been standardized in ITU G.711 [44] for 8 bits per sample at a sampling rate of 8000 Hz but the IETF RTP [45] and SDP standards allow the use of µ- and A-Law even at other sampling rates, for example at 48000 Hz. Thus, the use of logarithmic quantization (or PCM) allows the transmission of audio signals even with existing standards. The APT-X stereo codec has an algorithmic delay of 1.9 ms and a rate between 128 and 384 kbps but it is not available for free. Also, Fraunhofer's Ultra Low Delay Encoding [14] compresses stereo audio to 96 kbps with a frame size of 2.7 ms and an algorithmic delay of 5.4 ms. Again this codec is not available as open source. Recently, the CELT codec has been developed by J-M Valin et al. [16]. It is open source and has a very good quality vs. rate trade-off and very low algorithmic delays. We tested it at various sampling rates and frame sizes. We used it with an algorithmic delay of 150% of the reciprocal of the frame rate.

We tried to compare the performance of those coding schemes. However, we were not able to get a working implementation of ULD and APT-X. Thus, we asked a fellow researcher and a company to encoded and decode a large sample file containing multiple samples files. The large sample file contained the shorter samples used throughout this work but kept them separated by one second of silence. After getting back the en- and decoded large sample files, we removed aligned the file to the original and splitted it again into small files again. Next, we compared the original small samples with the degraded using the combined PEAQ metric. This step was done multiple times for different codecs and coding mode both in mono and stereo conditions.

It is a general consensus that PEAQ is not capable of comparing different codecs because the kind of distortions might be vary to a large extend. PEAQ might evaluate different kinds of distortions on different scales due comparing without proper subjective verification should be avoid.

Being aware of these facts, we still included PEAQ comparision results into this document knowing that they are in not means a suitable performance codec comparison but only an indication of quality. The PEAQ comparison results are shown in Figure 5.8 and 5.9.

The PEAQ-ODG ratings in the mono mode are clear. CELT v0.6 outperforms all other codecs at the tested rate vs. delay trade-offs. ULD is better as SBC if one considering just the bit rate and is equally good if looking at the more realistic gross rate. In the stereo mode, the results are not as clear. SBC seems to perform equally well as CELT v0.61. Further studies have to verify these results as they come unexpected.

Sample	*ODG avg.*	*ODG std.*	*Sample*	*PESQ-WB avg.*	*PESQ-WB std.*
refsmg01	-2.55	1.42	reftri	3.07	0.871
refsfe02	-2.49	1.41	refsfe02	3.30	0.883
refsme02	-2.48	1.45	rtde045	3.30	0.926
refsop01	-2.39	1.36	rtde040	3.33	0.849
refsme01	-2.38	1.49	rtde031	3.34	0.962
refveg02	-2.28	1.29	rtde035	3.36	0.866
refsfe01	-2.28	1.48	refsfe01	3.38	0.862
reftam	-2.22	1.42	refveg02	3.40	0.933
refryc	-2.18	1.44	refveg01	3.43	0.930
reftub	-2.17	1.67	refsop01	3.43	0.861
refglo	-2.14	1.46	kkoe025	3.45	0.980
refveg01	-2.12	1.44	hpte005	3.47	0.883
reftri	-2.10	1.44	refglo	3.47	0.724
refcla	-2.08	1.53	refsb1	3.48	0.962
refsb1	-2.00	1.46	ugae060	3.49	0.862
refpia01	-1.87	1.49	refsme01	3.49	0.810
refflu	-1.81	1.33	refsme02	3.49	0.865
hpte005	-1.80	1.36	refcla	3.50	0.814
refclv	-1.76	1.28	hpte015	3.50	0.937
average	-1.75	2.02	ugae078	3.50	0.818
refhrp	-1.67	1.35	hpte011	3.50	0.892
hpte011	-1.66	1.42	ugae068	3.51	0.895
kkoe019	-1.56	1.31	ugae051	3.51	0.826
rtde035	-1.46	1.35	refsmg01	3.52	0.782
ugae060	-1.46	1.29	*average*	3.52	0.728
rtde040	-1.46	1.32	kkoe026	3.55	0.905
ugae051	-1.42	1.35	kkoe019	3.55	0.778
reftpt	-1.41	1.28	hpte004	3.56	0.870
kkoe026	-1.38	1.28	refcas	3.56	0.842
rtde031	-1.38	1.32	reftpt	3.57	0.836
hpte004	-1.37	1.34	refryc	3.57	0.921
rtde045	-1.36	1.30	ugae063	3.57	0.855
refxyl	-1.35	1.26	kkoe023	3.58	0.830
ugae068	-1.35	1.28	reftam	3.60	0.823
kkoe023	-1.34	1.29	refflu	3.61	0.701
ugae078	-1.34	1.26	refhrp	3.64	0.834
hpte015	-1.34	1.26	refpia01	3.65	0.835
kkoe025	-1.27	1.29	reftub	3.71	0.881
refsna	-1.27	1.14	refsax	3.76	0.775
refsax	-1.25	1.14	refxyl	3.80	0.509
ugae063	-1.25	1.28	refsna	3.85	0.550
refcas	-1.23	1.12	refclv	3.89	0.490

Table 5.2: Loss concealment performance depending on sample content averaging over all tested PLC mode and loss rates. The left side shows the quality measured with averaged PEAQ approach. The right side gives the quality measured with PESQ-WB.

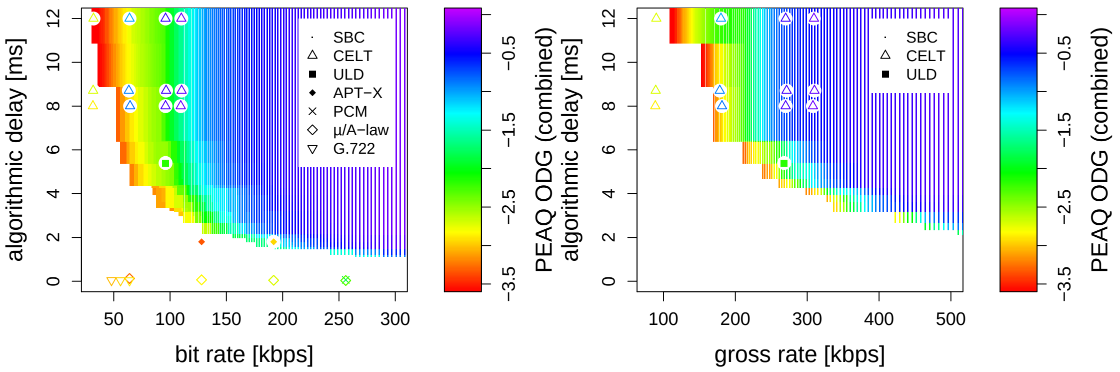


Figure 5.8: Mono: Objectively audio quality measured with combined PEAQ of samples encoded with different codecs and different coding modes. We display the ODG value versus algorithmic delay and bit/gross rate.

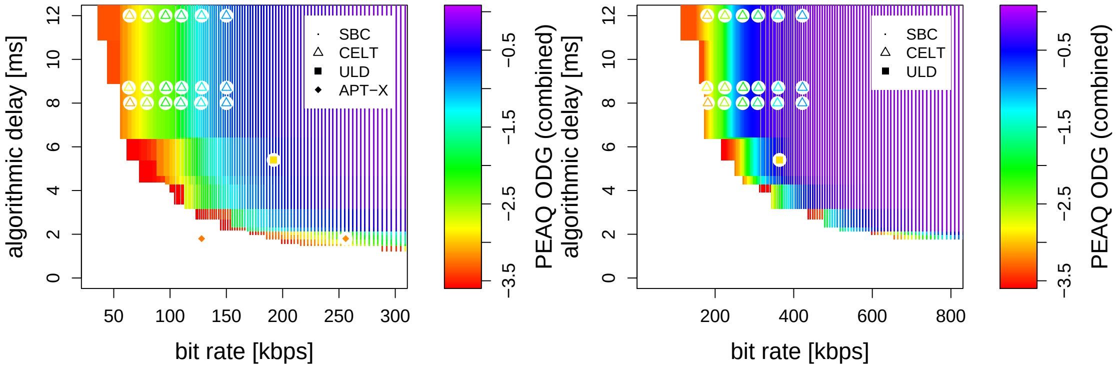


Figure 5.9: As Figure 5.8 but showing only the results for stereo coding.

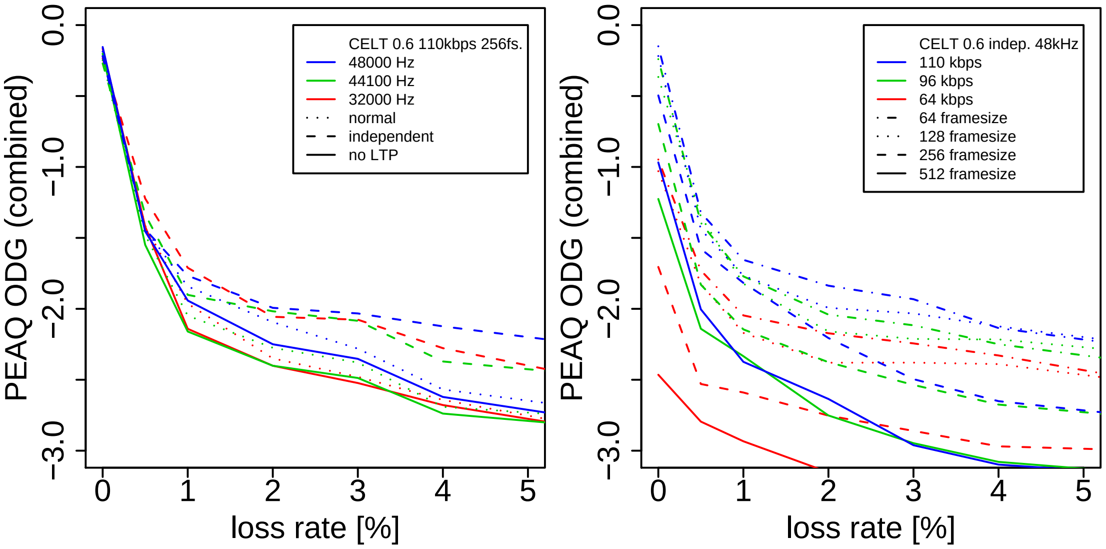


Figure 5.10: Performance of the CELT PLC for different coding modes and frame sizes

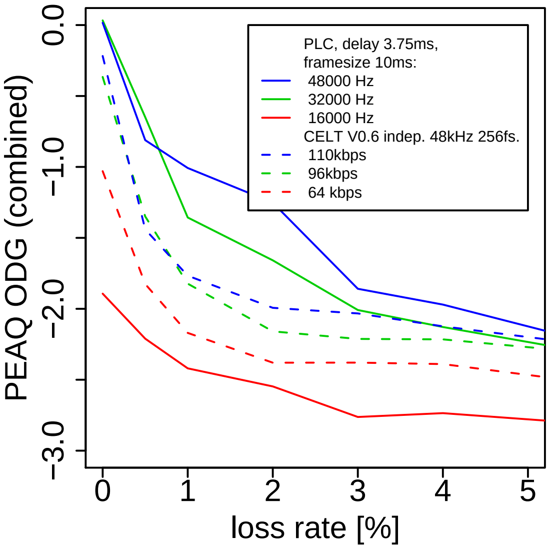


Figure 5.11: Performance of the PLC for different coding rates and algorithmic delays

6 Summary and Conclusion

This work contains the description of quite a number of research results. The first come from the comparison of subjective and objective audio quality assessments in case of distortions caused by SBC and our PLC. They show that the open source reimplementation of PEAQ basic has a couple of bugs. Also, Creusere extension to this software is currently not usable. Instead, the PEAQ reference implementation should be taken. The PEAQ advance version outperforms—as expected—the basic version slightly. However, the combination of both algorithms is even better. It is roughly comparable to a MUSHRA rating of a single human.

The second main contribution is in the area of design of a speech receiver indented for the Internet. We show a novel interface on how to connect a decoder to the surrounding system. This interface simplifies the design of a modern speech receiver. Also, it requires that the decoder supports more features beyond decoding, especially concealment of frame loss and playout time adjustments.

The third contribution is on the optimal usage of SBC for Internet transmission but also on wireless (Bluetooth) connections. Now, we know with transmission mode of SBC to chosen given bandwidth and delay constrains. For example, a Bluetooth A2DP device can operate more efficient. Also, the results show strength and weakness of SBC. Audio signals difficult to code with SBC are in general audio signals containing pure tones and stable harmonic series such as the harpsichord and the pitch pipe. On the other hand SBC is relatively good in coding audio signals with a high time resolution, e.g. castanets and applause.It is quite important to optimally select the right coding mode. If the encoder generates an compressed audio stream having too much bandwidth, the IP based transmission will experience packet losses, which fast degrades the audio quality. Also, if the encoder generates an audio stream at a lower than optimal bandwidth, a potential positive benefit of the coding efficiency fast diminishes.

We can conclude that making a codec ready for the Internet requires much more than writing a IETF payload specification. A good concealment algorithm to support the negative effect of various kind of distortion is important. Also, the classically used interface between codec and “the rest of the world” might lead to suboptimal results. Finally, instead of a perfectly working codec, it is of equal importance to optimally select the right coding mode. Otherwise, any optimization in the encoding performance is in vain.

Acknowledgments

We author would like to thank Franz de Bont, Marcel Holtmann, Brad Midgley, Henryk Plötz, Siarhei Siamashka and all the others to help to implement and debug BlueZ’s implementation of SBC. Also, we like to thank Matthias Bartl for conducting the subjective listening-only tests, Alexander Carôt for the ULD samples, and Tonstudio Berlin for APT-X samples.

Bibliography

[1] “Advanced audio distribution profile (A2DP) specification version 1.2,” http://www.bluetooth.org/, Apr. 2007, bluetooth Special Interest Group, Audio Video WG.

[2] 3GPP, “3rd generation partnership project;technical specification group services and system aspects;ip multimedia subsystem (ims);multimedia telephony; media handling and interaction,” technical specification 3GPP TS 26.114, Jun. 2009, version 8.3.

[3] *A High Quality Low-Complexity Algorithm for Packet Loss Concealment with G.711*, Recommendation G.711 Appendix I, ITU-T Std., Sep. 1999.

[4] *The E-Model, a Computational Model for Use in Transmission Planning*, Recommendation G.107, ITU-T Std., May 2000.

[5] C. Chafe, M. Gurevich, G. Leslie, and S. Tyan, “Effect of time delay on ensemble accuracy,” in *In Proceedings of the International Symposium on Musical Acoustics (ISMA-2004)*, Taipei, Taiwan, 2004.

[6] W. Woszczyk, J. Cooperstock, J. Roston, and W. Martens, “Environment for immersive multi-sensory communication of music using broadband networks,” in *In: 23. Tonmeistertagung VDT International Audio Convention*, Leipzig, Germany, 2004.

[7] A. Carôt and C. Werner, “Network music performance - problems, approaches and perspectives,” in *International School of new Media, Institute of Telematics, University of Lübeck*. Music in the Global Village - Conference, Sep. 2007.

[8] M. Hyder, M. Haun, and C. Hoene, “Placing the participants of a spatial audio conference call,” in *IEEE Consumer Communications and Networking Conference - Multimedia Communication and Services (CCNC 2010)*, Las Vegas, USA, Jan. 2010.

[9] R. J. Bernard, D. Y. Francois, and R. J. Yves, “Digital transmission system using subband coding of a digital signal.” European Patent, Publication number: EP0400755 (B1), Sep. 1995.

[10] ITU-R, “Method for objective measurements of perceived audio quality,” Recommendation BS.1387, Nov. 2001.

[11] *Perceptual Evaluation of Speech Quality (PESQ), an Objective Method for End-To-End Speech Quality Assessment of Narrowband Telephone Networks and Speech Codecs*, Recommendation P.862, ITU-T Std., Feb. 2001.

[12] C. Creusere, K. Kallakuri, and R. Vanam, “An objective metric of human subjective audio quality optimized for a wide range of audio fidelities,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 129–136, Jan. 2008.

[13] G. Schuller, B. Yu, D. Huang, and B. Edler, “Perceptual audio coding using adaptive pre- and post-filters and lossless compression,” *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 6, pp. 379–390, Sep. 2002.

[14] J. Hirschfeld, J. Klier, U. Kraemer, G. Schuller, and S. Wabnik, “Ultra low delay audio coding with constant bit rate,” in *AES Convention 117*, no. 6197, October 2004.

[15] [Online]. Available: http://www.aptx.com/

[16] Xiph.Org Foundation, “The CELT ultra-low delay audio codec,” http://www.celt-codec.org/, Mar. 2009.

[17] F. Bont, M. Groenewegen, and W. Oomen, “A high-quality audio coding system at 128 kb/s,” in *AES 98th Convention*, no. 3937, Paris, February 1995.

[18] ISO/IEC, “Iso/iec 11172-3:1993:coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s – part 3: Audio,” standard.

[19] K.-E. Christensen and E. Sorensen, “A fast algorithm for polyphase quadrature filters,” *IEEE Transactions on Signal Processing*, vol. 42, no. 12, pp. 3513–3515, 1994.

[20] D. Hermann, R. Brennan, H. Sheikhzadeh, and E. Cornu, “Low-power implementation of the bluetooth subband audio codec,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP’04)*, vol. 5, May 2004, pp. V–365–8 vol.5.

[21] M. Holtmann. (2009, Apr.) BlueZ - official linux bluetooth protocol stack. [Online]. Available: http://www.bluez.org/

[22] C. Hoene, “Conformance and quality tests of the SBC audio codec implementation in the linux kernel,” online, Dec. 2008. [Online]. Available: http://www.nexgenvoip.org/

[23] R. Ramjee, J. F. Kurose, D. F. Towsley, and H. Schulzrinne, “Adaptive playout mechanisms for packetized audio applications in wide-area networks,” in *13th Proceedings IEEE INFOCOM ’94*, Toronto, Canada, Jun. 1994, pp. 680–688.

[24] S. B. Moon, J. Kurose, and D. Towsley, “Packet audio playout delay adjustments: performance bounds and algorithms,” *ACM/Springer Multimedia Systems*, vol. 27, no. 3, pp. 17–28, Jan. 1998.

[25] N. Kitawaki and K. Itoh, “Pure delay effects on speech quality in telecommunications,” *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 4, pp. 586–593, 1991.

[26] R. Steinmetz, “Human perception of jitter and media synchronization,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 61–72, 1996.

[27] Y. J. Liang, N. Fï¿œrber, and B. Girod, “Adaptive playout scheduling and loss concealment for voice communication over IP networks,” *IEEE Transactions on Multimedia*, vol. 5, no. 4, pp. 532–543, Dec. 2003.

[28] F. Liu, J. Kim, and C.-C. J. Kuo, “Adaptive delay concealment for internet voice applications with packet-based time-scale modification,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP ’01)*, vol. 3, May 2001, pp. 1461–1464.

[29] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi, “Detailed analysis of skype traffic,” *Multimedia, IEEE Transactions on*, vol. 11, no. 1, pp. 117–127, Jan. 2009.

[30] ITU-T, “Specification and description language (sdl),” Z.100, Aug. 2002.

[31] C. Hoene, H. Karl, and A. Wolisz, “A perceptual quality model intended for adaptive VoIP applications,” *International Journal of Communication Systems, Wiley*, Aug. 2005.

[32] K. Vos, S. Jensen, and K. Soerensen, “Silk speech codec,” IETF Internet-Draft draft-vos-silk-00.txt, Jul. 2009, work in progress.

[33] C. Hoene, K. Clüver, and J. Weil, “An architecture for a next generation voip transmission system,” in *Praxis der Informationsverarbeitung und Kommunikation*, vol. 30, no. 2, April-June 2007. [Online]. Available: http://dx.doi.org/10.1515/PIKO.2007.76

[34] V. Vilaysouk and R. Lefebvre, “A hybrid concealment algorithm for non-predictive wide-band audio coders,” in *AES 120th Convention*, A. 120th Convention, Ed., Paris, France, May 20-23 2006.

[35] D. R. Reddy, “Pitch period determination of speech sounds,” *Commun. ACM*, vol. 10, no. 6, pp. 343–348, 1967.

[36] ITU-R, “Method for the subjective assessment of intermediate quality levels of coding systems,” Recommendation BS.1534-1, Jan. 2003.

[37] M. Bartl and C. Hoene, “An open-source softphone for musicians playing over IP,” ITU-T Workshop ’From Speech to Audio: bandwidth extension, binaural perception’, Lannion, France, Sep. 2008.

[38] T. Thiede, W. C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, and C. Colomes, “Peaq - the itu standard for objective measurement of perceived audio quality,” *JAES*, vol. 48, no. 1/2, pp. 3–29, Feb. 2000.

[39] W. C. Treurniet and G. A. Soulodre, “Evaluation of the ITU-R objective audio quality measurement method,” *JAES*, vol. 48, no. 3, pp. 164–173, Mar. 2000.

[40] R. Huber and B. Kollmeier, “Pemo-q – a new method for objective audio quality assessment using a model of auditory perception,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 6, pp. 1902–1911, Nov. 2006.

[41] V. Grancharov and A. Taleb, “Analysis of PESQ’s applicability in predicting the quality difference between alternative implementations of the g.722.1fb coding algorithm,” ITU-T Study Group 12 - Contribution 163, May 2008.

[42] E. Hellerud, P. Svensson, and J. E. Voldhaug, “Evaluation of packet loss distortion in audio signals,” in *AES Convention: 120*, no. 6855, May 2006.

[43] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice Hall Professional Technical Reference, 1990.

[44] *Pulse Code Modulation (PCM) of Voice Frequencies*, Recommendation G.711, ITU-T Std., Nov. 1988.

[45] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: a transport protocol for real-time applications,” RFC 3550, IETF Network Working Group, Jul. 2003.