Comparative Study of Low-Latency Audio Codecs

Purbaditya Bhattacharya
Helmut Schmidt University / University of the Federal Armed Forces Hamburg
10 PUBLICATIONS **17** CITATIONS

HELMUT SCHMIDT
UNIVERSITÄT

Universität der Bundeswehr Hamburg

Comparative Study of Low-Latency Audio Codecs

Studienarbeit

by

Purbaditya Bhattacharya

Start date:	1.5.2013
End date:	26.5.2014
Supervisor:	Dipl.-Ing. Marco Fink
Supervising Professor:	Prof. Dr.-Ing. habil. Udo Zölzer

Abstract

Transmission of Audio over IP (AoIP) for real time streaming audio applications and for networked music performance requires low latency audio coding. Present audio coding standards like Mp3, Vorbis introduce high amount of coding delay while low latency speech codecs cannot produce high audio quality. This project work focuses on two low latency audio codecs called Opus and BCDPCM and their comparative study. A test framework is built for simulating the behavior of the codecs under different input configuration and comparison of the the coded audio quality. Supporting scripts are also written for faster performance of the test procedure and a graphical interface is built. A theoretical study of other attributes of the codecs are also done for comparison.

Statement

I, Purbaditya Bhattacharya, student of Information and Communication systems at the Technical University of Hamburg-Harburg, matriculation number 21152813, hereby state that this work has been prepared by myself and with the help which is referred within this thesis.

Hamburg, 26/5/2014

Contents

List of Figures

List of Tables

List of Symbols

Symbol	**Description**
ms	Milliseconds
kHz	Kilohertz
kbps	Kilobits per second
km	Kilometers
w	Window function
n	Sample index
L	Window length
Q	Quantizer
N	Frame size/band size
E	Energy
M	MDCT
x	Input signal
$\tilde{x}$	Predicted input signal
y	Output signal
$\tilde{y}$	Predicted output signal
e	Unquantized error
$\bar{e}$	Quantized error
$\hat{e}$	Dequantized error
P	predictor
f_i	Forward prediction error for i_{th} stage
b_i	Backward prediction error for i_{th} stage
f_0	Initial forward prediction error
b_0	Initial backward prediction error
m	Filter stage
p	Maximum prediction filter order
k	PARCOR coefficient/reflection coefficient
μ_i	Gradient weight at i_{th} stage of lattice filter
$\hat{\mu}$	Initial gradient weight
σ_m^2	Error power variance
α,β	Damping factors/leakage factors
$n_f(i)$	Normalization factor for i_{th} block
r_b	Bit rate

l_B	Block length
p_l	Packet loss rate
$\hat{f}_s$	Effective sampling rate
g	Gain
a	Pre and post filter coefficients
T	Pitch period
K	Number of pulses
D	Dimension
b	Number of bits
b_e	Number of bits per sample for the residual signal
b_n	Number of bits per sample for the Normalization factor
r	Recalculation factor
r_{mod}	Modified recalculation factor
N_f	Total number of frames
x_i	Factors
tp	Taps

Chapter 1

Introduction

Internet over the years became the most popular medium for communication. It is a packet switched network and it supports a wide variety of applications ranging from sending messages to streaming live feed in high definition quality. It has evolved and is still evolving into a highly scalable structure. Internet also offers flexibility in terms of its underlying protocols.

Traditional circuit switched networks, like the telephone network have a dedicated channel and provide end to end voice communication. But communication across the globe via telephony offer challenges in terms of scalability, availability, cost, and resource usage. Today advances in applications for voice over IP (VoIP), like Skype, has replaced the usage of public telephony particularly for long distance communication.

Since internet already provides the opportunity for communication between multiple entities in real-time, sharing live audio data over IP is in popular demand. But internet does have its drawbacks. It is unreliable in terms of

- Network Capacity
- Network Latency
- Packet Loss

Uncompressed Compact Disk (CD) quality stereo audio requires 1.41 Mbps bit rate and this data rate scales with the number of channels. In internet the transmission medium is shared among multiple users which limits available bandwidth for individual applications.

Unlike voice, audio is more affected by the above issues according to the perception of a listener. In a two way live audio performance, for synchronization of the parties involved, it is necessary that the delay is very small and the quality of audio is good. In this respect an audio encoder and decoder should maintain transparency in audio quality yet introducing very low latency, low computational complexity and good packet loss robustness. Clearly a compromise has to be agreed upon.

In this topic the audio codecs of interest are Opus codec and Block Companded Differential Pulse Code Modulation (BCDPCM) codec. The goal of this thesis is to simulate and compare the performance of these codecs in terms of audio quality and provide a theoretical analysis of the latency and the computation cost introduced.

Chapter 2 of this thesis will provide a description of the codecs. The test setup for analyzing the behavior of these codecs is provided in chapter 3. A theoretical analysis of codec complexity and latency is done in chapter 4. The results and corresponding explanation will constitute chapter 5. Chapter 6 will contain the Conclusion.

Chapter 2

Fundamentals

In this chapter a brief description of the Opus and BCDPCM codec are provided. Opus [1] is a transform codec based on Modified Discrete Cosine Transform (MDCT) that reproduces the spectrum of the input audio. On the other hand BCDPCM [2] is a waveform codec that tries to reconstruct the time domain input signal. The quality of the coded audio is measured by an application called Perceptual Evaluation of Audio Quality (PEAQ) [7].

2.1 Audio Codec

A selection of lossy audio codecs is available but their suitability for transmission over IP is questionable. Audio compression algorithms in Vorbis or MP3 introduce very high delay due to long frame size which is not suitable for AoIP. While AAC format introduces variable amounts of delay, a low delay version called AAC-LD introduces delay as low as 20 ms but it requires comparatively higher bit rate to produce transparent audio quality. Another audio compression format called aptX is capable of producing very low delays comparable to BCDPCM but operates with higher bitrates. Thus Opus codec is suitable for music streaming and remote music jamming applications while with BCDPCM a low latency bidirectional communication can be achieved.

2.1.1 Opus

Opus performs lossy compression supporting bitrates of 6-512 kbps and can encode signals ranging from narrowband speech (8 kHz) to wideband audio (48 kHz). It operates with both constant and variable bitrates as required and uses distinct frame sizes from 2.5 to 60 ms.

2.1.1.1 Modes of operation

The Opus codec uses two coding technologies to encode audio and speech signals. Audio and high fidelity speech signals are encoded with the Celt codec while normal voice is encoded with the Silk codec, used in Skype. Depending on the content of the audio Opus uses both, but the user have the option to be selective. The block scheme of the Opus codec is provided in Fig. 2.1. Opus offers a music mode which only encodes audio and a non-music mode or speech mode where speech is encoded. The music mode is used and described in this work.

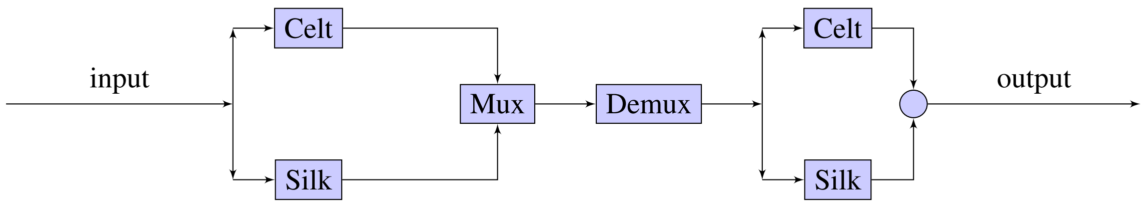


Figure 2.1: Block scheme of the Opus codec

2.1.1.2 Opus Celt

The Celt codec is a low delay codec used to encode music or high fidelity speech. It is based on MDCT using short frames between 2.5 to 20 ms. It operates at a default sampling rate of 48 kHz. The Celt codec take into account, the perceptual properties of human ear to different kinds of audio signal and the content of the audio signal. The frequency band structure encoded by celt resemble the critical band structure of the human auditory system.

Block transforms with short window size results in leakage of energy into neighboring frequencies. Non-overlapping blocks due to discontinuities at their edges suffer from the above and result in audible artifacts. Thus Celt, like most audio codecs, uses a lapped transform where each block is windowed and these windows overlap, termed as look-ahead. The overlap size is 2.5 ms. Celt uses a flat top window. The overlapping part of the window illustrated in Fig. 2.2 is given by [3]

$$w(n) = \sin[(\pi/2). \sin^2(\pi(n+0.5)/2L), \tag{2.1}$$

where, $w(n)$ denotes a window function, n denotes number of samples, L denotes the length of overlap region.

Lapped transform introduces errors in the transformed output called time-domain aliasing. At the Celt decoder the inverse MDCT (IMDCT) is performed to retrieve the time-domain signal. Two subsequent blocks are then overlapped and added canceling the aliasing. This operation is termed as time domain alias cancellation (TDAC). In the absence of any quantization error this results in perfect reconstruction.

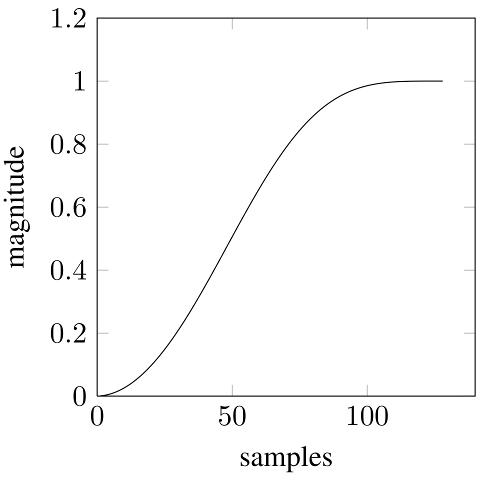


Figure 2.2: Overlapping part of the window function

The block diagram of the Celt encoder and decoder is represented in Fig. 2.3. The PCM input is pre-processed by the pre-fil block and then divided into frames and windowed (W). The pre-prossesing tasks involve modifying the input signal to cope against the effects of spectral leakage. Each window is overlapped with a fixed duration of the next windowed frame and are transformed to the frequency domain in the block M. The spectrum is divided into critical bands and band energy is computed for each (Band E). The band energy is coarsely quantized in Q_1 block and the quantization error is quantized in block Q_2 with a finer resolution. The spectral coefficients are normalized by the respective band energy to produce an unit vector which is vector quantized in block Q_3.

In the decoder the quantized symbols are dequantized by blocks Q_1^{-1}, Q_2^{-1}, Q_3^{-1} and the frequency domain results are transformed back to time domain in M^{-1} block. W_{OLA} block performs an windowed overlap and add of two consecutive time indexed transformed frames to cancel time domain aliasing effects. Post-fil block performs post filtering tasks that includes inverting the effects of the pre-fil block.

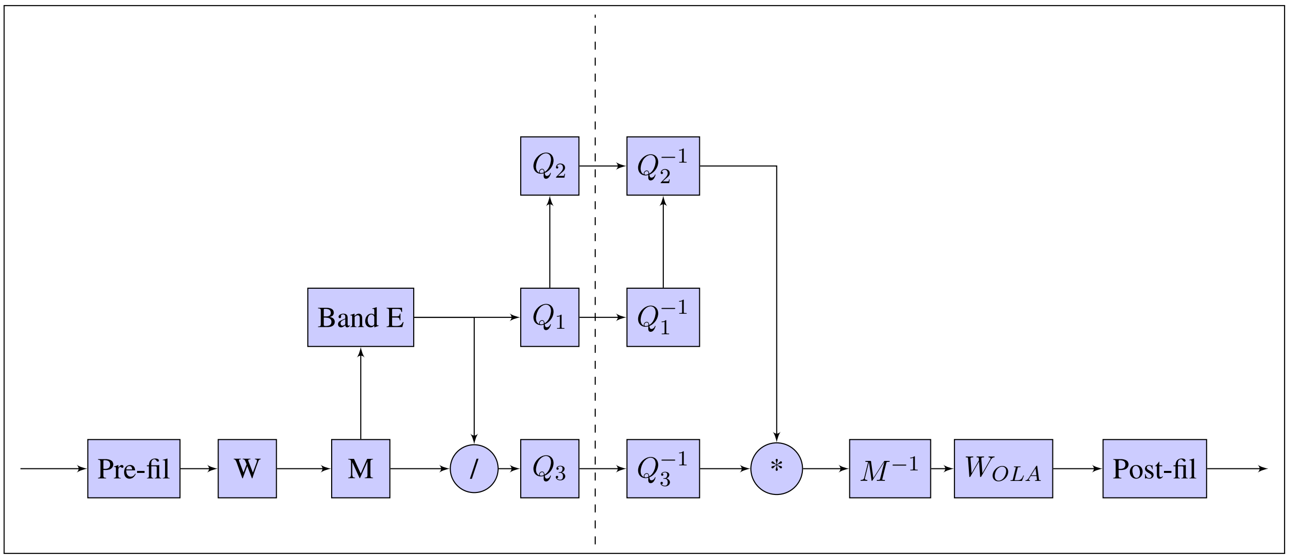


Figure 2.3: Simplified diagram of the Celt codec

2.1.1.3 Quantization

Celt quantizes the band energies and the spectral coefficients separately. Initially it calculates energies of individual bands and employs a coarse-fine quantization strategy. Celt focuses on preserving the band energies by encoding them with high resolution.

- In coarse quantization (Q_1) a prediction of the current band energy is made based on the previous band and the previous frame [3].

  $$A(z_l,z_b) = (1 - \alpha z_l^{-1}).\frac{(1 - z_b^{-1})}{(1 - \beta z_b^{-1})}, \tag{2.2}$$

  where l is the frame index, b is the band index and α and β are prediction coefficients.

  The predicted band energy is quantized with a 6 dB resolution.

- In fine quantization (Q_2) the quantization error due to Q_1 is quantized with a variable resolution depending on the availability of bits.

The spectral coefficients are divided by the respective unquantized band energy and thus the band is normalized producing a N-dimensional unit vector, N being the number of spectral coefficients. This vector is quantized with a specific vector quantization technique called Pyramid Vector Quantization (PVQ) [3,4]. The codebook used is of fixed size and quantization of normalized bands results in high data compression (less than 1 bit per dimension)

2.1.1.4 Bit allocation

The Celt encoder and decoder use the same bit allocation function to distribute bits among bands. Each band is allocated with a fixed number of bits. The low-frequency bands receive more bits than the higher frequency bands. Apart from this static bit allocation there are two other mechanisms that offer flexibility [3].

- Allocation Tilt: A fraction of bit allocation is done as a function of band index.

- Band Boost: Specific bands with higher energy content receive more bits than the others.

Transients in audio (percussive instruments) introduce an artifact called pre-echo, where the listener hears a particular sound or effect before it has actually occurred, because quantization noise is spread across the entire frame as shown in Fig. 2.4 where the initial samples before the transient attack, are affected as well.

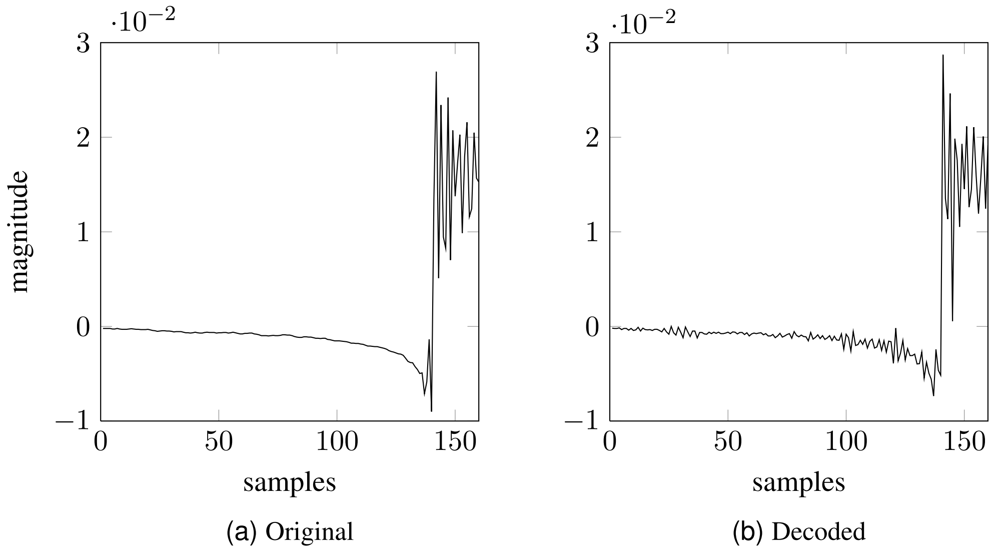


Figure 2.4: Pre-echo in Castanet

This issue require MDCT blocks to be divided into shorter blocks, resulting in increased spectral leakage which is difficult to encode. More bits are allocated to the corresponding bands. The bit allocation however is adjusted to meet the input bitrate. Certain bands might get no bits particularly at lower bitrates. The bit allocation curve examples for different bitrates are shown in Fig. 2.5.

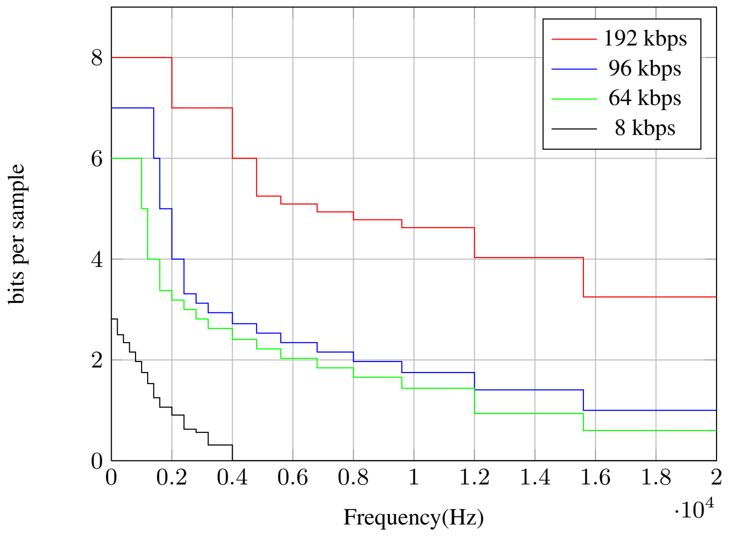


Figure 2.5: Bit allocation for different input bitrates

2.1.1.5 Pre and post processing

Short frame size causes spectral leakage. Celt employs a pre-emphasizing filter to attenuate low frequencies so as to reduce the amount of leakage in higher frequencies. The de-emphasizing filter in the decoder reverses the above.

A pre and post-filter is used to mitigate effects of high spectral leakage in nearly periodic signals due to short overlap windows in MDCT (low frequency resolution). Tonal signals have strong harmonics and spectral leakage due to low overlap short windows result in high quantization noise. The widely spaced harmonics provide less masking and hence filtered with a pitch-based comb filter. The post-filter reverses the above.

The bit allocation strategies, the internal changes in framesize to meet the time frequency resolution requirement of different types of audio signal, and the pre and post processing task ensure that the perceptual properties of the audio signal are taken into account.

2.1.1.6 Error Robustness

Celt is robust to packet loss. It uses packet loss concealment (PLC)[5] strategy based on pitch prediction to suitably conceal an actual packet loss. On the occasion of a packet loss of very low magnitude, the decoder replaces the lost packet by noise like data, when pitch based concealment is costly with respect to the percentage of loss or the significance of the lost packet. Generally, the lost packet will be replaced by a scaled or adjusted copy, one pitch period apart.

2.1.2 BCDPCM

The block companded differential pulse code modulation codec is a low latency waveform codec. It employs linear adaptive prediction and block-based quantization for encoding audio data. It achieves compression by reducing redundancies in time-domain audio signal.

The block diagram of BCDPCM codec is provided in Fig. 2.6. The predictor estimates the future sample value $\tilde{x}(n)$ based on the present sample x. The resulting prediction error e, also termed as the residual, is quantized. As the prediction coefficients adapt recursively to new input samples the residuals reduce to very low values with low mean deviation. They are normalized and sent to the decoder. The decoder runs the reverse prediction algorithm, and provided there is no packet loss, achieves appreciable reconstruction.

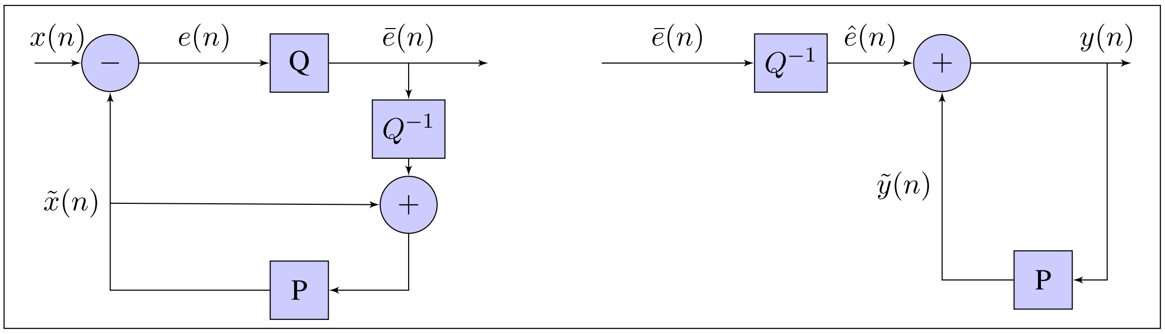


Figure 2.6: Block diagram of BCDPCM Encoder and Decoder

2.1.2.1 Linear Adaptive Predictor

The predictor is realized in a lattice structure of order p as shown in Fig. 2.7 and the following set of equations define the prediction steps.

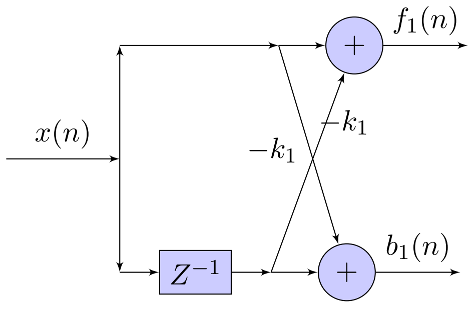


Figure 2.7: A single stage of the prediction filter

$$f_m(n) = f_{m-1}(n) - k_m b_{m-1}(n-1), \tag{2.3}$$

$$b_m(n) = b_{m-1}(n-1) - k_m f_{m-1}(n) \quad with\, m = 1, \ldots, p, \tag{2.4}$$

where $f_m(n)$ and $b_m(n)$ are the forward and backward prediction errors, $k_m(n)$ is a reflection coefficient called normalized partial correlation (PARCOR) coefficient, p is the prediction order and n is the sample index. The output $f_p(n)$ is the prediction error or residual and the filter inputs $f_o(n)$ and $b_o(n)$ are equal to the input sample.

The reflection coefficients are recursively updated by [2]

$$k_m(n+1) = k_m(n) + \mu_m(n)(f_m(n)b_{m-1}(n-1) + f_{m-1}(n)b_m(n)) \quad ; \; m = 1, \ldots, p, \tag{2.5}$$

where $\mu_m(n)$ is gradient weight given by

$$\mu_m(n) = \hat{\mu}/(\sigma_m^2(n) + \sigma_{min}^2) \quad m = 1, \ldots, p, \tag{2.6}$$

$$\sigma_m^2(n) = (1-\hat{\mu})\sigma_m^2(n-1) + \hat{\mu}(f_{m-1}^2(n) + b_{m-1}^2(n)) \quad ; \; m = 1, \ldots, p. \tag{2.7}$$

The above algorithm is called gradient adaptive lattice (GAL) algorithm.

2.1.2.2 Error Robustness

In case of transmission errors the decoder will produce a wrong estimate of all coefficients and this will carry the erroneous results across the future estimated samples. The effects can be minimized by a error concealment strategy or by sending the coefficients calculated at the encoder. But this would also increase the complexity and the transmission overhead of the system.

Currently a constant leakage factor α and β are used as damping factors that prevent the decoder from producing extensive erroneous output [6]. The equations (2.3) and (2.4) are updated as follows

$$f_m(n) = f_{m-1}(n) - k_m \alpha b_{m-1}(n-1) \quad m = 1, \ldots, p, \tag{2.8}$$

$$b_m(n) = \alpha b_{m-1}(n-1) - k_m f_{m-1}(n) \quad m = 1, \ldots, p. \tag{2.9}$$

The equation (2.5) for calculating the reflection coefficient is updated as follows

$$k_m(n+1) = k_m(n) + \mu_m(n)(f_m(n)\beta\alpha b_{m-1}(n-1) + f_{m-1}(n)b_m(n)) \quad m = 1, \ldots, p. \tag{2.10}$$

2.1.2.3 Normalization Factor

BCDPCM calculates a normalization factor for a block of residual samples recursively. The block size used here is 16. The steps to calculate normalization factors for sample index n and block index i are given below [2]

1. At beginning of the block the residual $e(n)$ is calculated in the prediction process.
2. If $e(n)$=0, then normalization factor $n_f(i)$ is a predefined small real number else $n_f(i)$=$e(n)$.
3. For any future sample within the block, if $e(n)$ is greater than current value of $n_f(i)$, then $n_f(i)$=$e(n)$, the recalculation counter rc is incremented, and the prediction process is run again from the beginning of the block with the new value of $n_f(i)$ as an input.
4. If the above process repeats beyond a maximum number of recalculation(rc>11) then no further recalculation is done and the latest value of $n_f(i)$ is used till block end.
5. For any future sample within the block, if $e(n)$ is less than current $n_f(i)$ till block end then there will be no recalculation and the current value of $n_f(i)$ will be the final one.
6. After the end of the block, the entire process will be repeated for a new block and rc will reset.

2.1.2.4 Quantization

The normalized residual signal is quantized by a Max-Lloyd Quantizer with 3 or 4 bits/sample and the normalization factors are quantized by a scalar quantizer with 6 bits/16 samples or 0.375 bits/sample.

A Max-Lloyd quantizer is a non uniform quantizer using fixed size codebooks corresponding to pre-defined bits allocated per sample. It is adaptive in terms of normalized residuals which are computed adaptively according to the previous section.

The normalization factors have to be sent over to the decoder resulting in 9.8 to 12.5 % percent overhead.

2.2 Measurement of Audio Quality

Subjective tests for measuring audio quality are very reliable. A keen listener might be able to decide on the difference in audio quality between two versions accurately but the method can be time consuming. Perceptual Evaluation of Audio Quality (PEAQ) is an objective audio quality test tool that compares the original and the decoded audio and provides a grade to the decoded version. A basic block scheme for evaluation of quality of audio by PEAQ, is provide in Fig. 2.8.

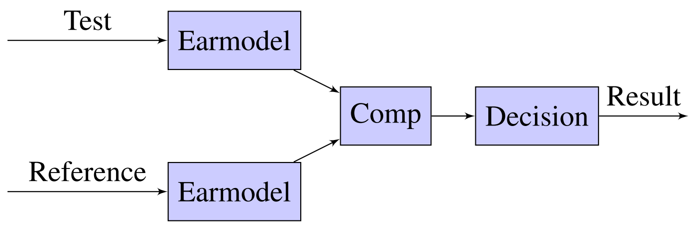


Figure 2.8: Basic block scheme for audio quality evaluation using PEAQ

The ear model block replicates the human ear and the excitation patterns created in the human ear by a sound. A psycho-acoustic analysis is done and required parameters are calculated for both the reference and the decoded audio. The results are compared and the deviation of values from the reference are recorded. The decision block is a cognitive model that decides on what grades to assign depending on mean parametric deviation patterns.

The ear model performs time-frequency conversions and psychoacoustic analysis. The ear model output are processed to analyze some absolute features of the reference and decoded audio and compared. This results in a set of variables at different stages of the comparison and decision model referred to as model output variables (MOV) [7]. These variables include:

- Absolute and comparative measure of temporal envelopes at the auditory filter.
- Measure of loudness of noise.
- Measure of noise-to-mask ratio (NMR).
- Absolute and comparative measure of signal bandwidth.
- Detection probability of perceptible difference between reference and decoded signals.
- Measure of harmonic distortion in decoded signal.

PEAQ requires a reference wav file and a file under test as its input.

Based on the output variable measures a decisive grade is given to the signal under test with respect to the reference.

The grades are listed in Table 2.1

Description	**ODG**
Imperceptible	0. 0
Perceptible, not annoying	-1. 0
Slightly annoying	-2. 0
Annoying	-3. 0
Very annoying	-4. 0

Table 2.1: Objective Difference Grade

The grades range from bad (-4.0) to good (0.0). A good grade implies that the decoded audio signal is qualitatively close to its reference while a decreasing grade means deterioration from reference.

Chapter 3

Experimental Setup

The basic idea behind the test setup is to run the codecs over a set of audio files under different configurations of allowed bitrates, frame sizes, and packet loss rates. The output audio files are then compared with their reference input for quality evaluation.

3.1 Test Arrangement

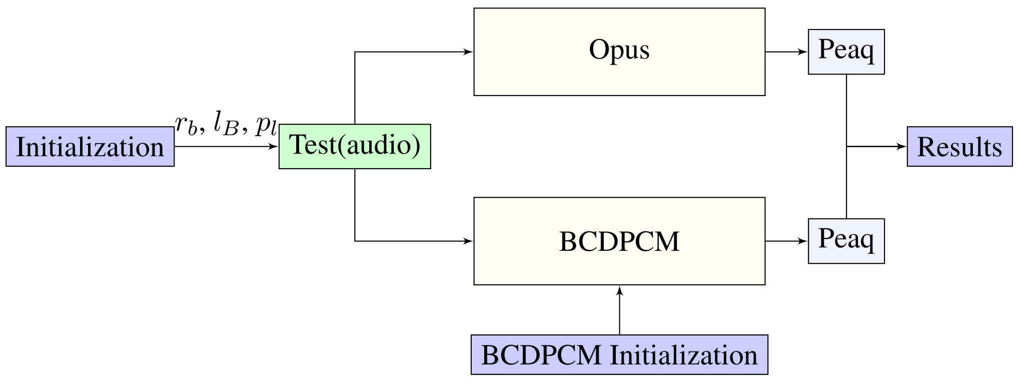


Figure 3.1: Block scheme of test procedure

The test block receives initial parameters bitrate (r_b), blocklength (l_B), and packetlossrate (p_l) from the initialization block including an array of audio files. It subsequently calls the BCDPCM and Opus block to encode and decode an audio file and store the output. This is followed by an evaluation of the coded audio quality with respect to the original and the graphical results are displayed. The process repeats for each audio file and is illustrated in Fig. 3.1.

3.1.1 Initialization and Test

In the initialization script all the reference stereo audio files are converted to mono, scaled to avoid clipping during the coding process and saved. The control parameters r_b, l_B, p_l are defined and initialized with valid values.

The test pseudo code and its explanation is given below.

```
Init; % Initializes the vectors of audio files, bitrates, frame
   size and packet loss percentages %with values provided

%%%% MAIN TEST LOOP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for id=1:length(fileIdx) % Iteration over the audio files

reference=file(id);

for b=1:length(blocklength) % Iteration over frame size
for p = 1:length(packetlossrate) % Iteration over packet loss
   percentage
for r = 1:length(bitrate) % Iteration over bitrates

%%%% CALLS OPUS CODER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OPUSencdec(reference,outputfileOPUS,opus_args(b,p,r),other_args);

%%%%% CALLS PEAQ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

peaqScoreOPUS(r,p,b)  = peaqInterface(reference,outputfileOPUS);

%% DATA IMPORTED FROM A OPUS RUN FOR BCDPCM INPUT %%%%%%%%%

IMPORT_FROM_OPUS(id,p,r)=importdata(opussample); % import data
   saved during OPUS run

%%% included in other_args of BCDPCM coder %%%
index(id,p,r)=find((IMPORT_FRM_OPUS(id,p,r))~=0);

if packetlossrate(packetloss)~=0
    lost_packets(id,p,r)=index(id,p,r); % Find the lost packet
       indices in OPUS and store them
else
    lost_packets(id,p,r)=0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% CALLS BCDPCM CODER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BCDPCMencdec(reference,outputfileBCDPCM,BCDPCM_args(b,p,r),
   other_args);


%%%% CALLS PEAQ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
peaqScoreBCDPCM(r,p,b)  = peaqInterface(reference,outputfileBCDPCM);

end
end
end

peaqOP{id}=peaqScoreOPUS;
peaqBC{id}=peaqScoreBCDPCM;
end
Visualization % Check results graphically
```

3.1.2 Opus Codec

Opus is available in the form of source code and libraries named **libopus** written in C code. The latest version is libopus-1.1. It also provides a command line utility **opus-tools**. A user can provide parameters accepted by opus in the command line.

Libopus-1.1 has been used along with opus-tools-0.1.2. Opus encoder converts a .wav audio file to bitstream with .opus extension. Control parameters offered by opus encoder and decoder used in the test are given below:

- Encoder
  - bitrate: Controls target bitrate from 6 to 256 kbps per channel.
  - sampling rate: Controls sampling rate (raw input).
  - mode: Selects music or speech mode.
  - vbr/cvbr/cbr: Selects between variable bit rate, constrained variable bitrate, and constant bitrate.
  - complexity: Input encoding computational complexity.
  - framesize: Selects framesize, in ms. Distinct values are possible-2.5, 5, 10, and 20.
  - downmix: Downmix stereo audio to mono or downmix to stereo if number of channel is greater than 2
- Decoder
  - sampling rate: Controls sampling rate (raw input).
  - gain: Adjusts the output volume
  - packet loss: Accepts packet loss in percentage.

The packet loss concealment (PLC) cannot be controlled by any parameter. The user has to make changes in the source code to get the desired. The Opus script accepts parameters from test script and runs the encoder and decoder. In case of packet loss, the index of lost opus packets and total number of packets are imported to matlab for later usage. Opus output exhibits some lag with reference to the input.

3.1.3 BCDPCM Codec

The BCDPCM codec is written in Matlab with distinct modules for encoder and decoder. Like Opus, BCDPCM accepts control parameters and audio file data from the test module but it also requires some extra parameters used by its submodules. The block scheme for BCDPCM module is given in Fig. 3.2.

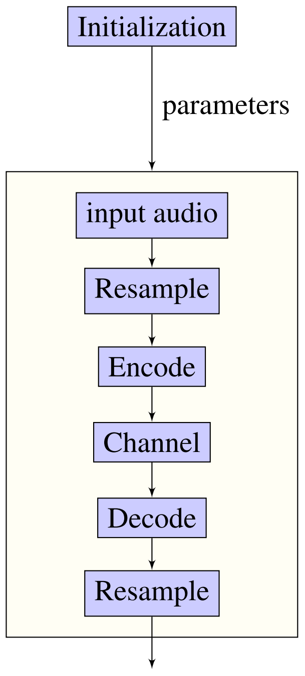


Figure 3.2: Block scheme of BCDPCM script

The initialization function provides companding block length, prediction filter order, initial values of prediction filter coefficients, number of bits for quantization, codebooks, and other required variables.

For BCDPCM number of bits (b_e and b_n) required for quantization of prediction residual and normalization factor respectively and the block size (l_B) for calculation of normalization factors are predefined. So depending on the bitrate (r_b) in kbps the effective sampling rate $\hat{f}_s$ is given by

$$\hat{f}_s = \frac{r_b}{b_e + (\frac{b_n}{l_B})}. \tag{3.1}$$

Thus the audio file which is sampled by default at 44.1 kHz needs to be resampled by a factor of

$$\frac{\hat{f}_s}{44.1}. \tag{3.2}$$

The re-sampled data will be passed as input to the encoder along with control parameters and initialization variables. The encoder transfers the quantized residuals along with overhead data through packet loss simulation. The Channel block is responsible for packet loss in BCDPCM.

Packet loss is simulated in two ways:

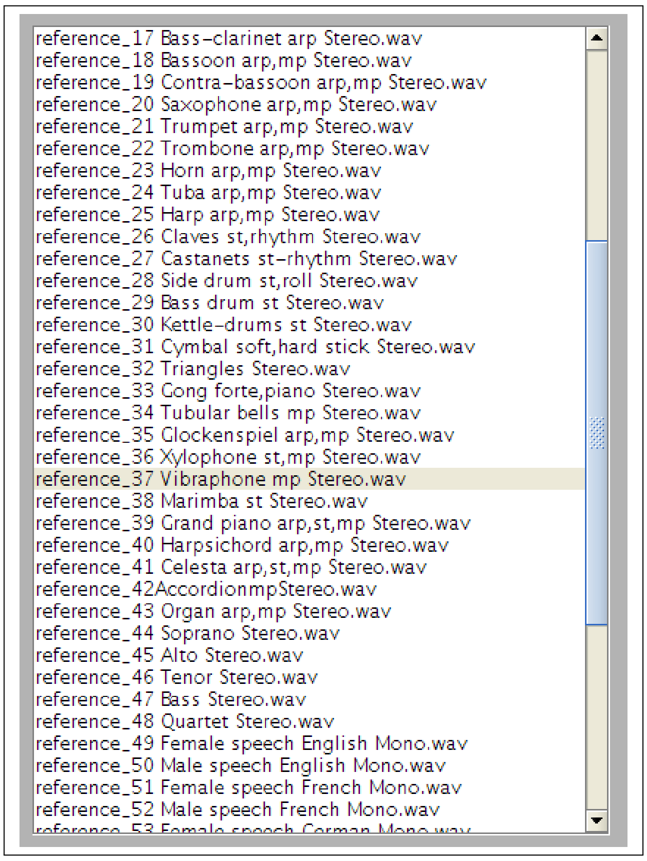
- the total number of output samples are divided into same number of packets as in opus and the same packets that are lost in Opus are lost in BCDPCM as well. This is done by importing the data about indices of lost packets from Opus. So, for higher bit rates BCDPCM loses packets of greater size in terms of bits per packet.

- the output samples are divided into packets of the same size as in opus (samples per packet) and packets are lost either randomly for the given packet loss percentage.

The data is decoded, resampled, and written to a .wav file

3.1.4 User Interface

A graphical user interface (GUI) is made for easier interaction involving initialization parameters. The user can load the audio files from respective location and select them from a list as shown in Fig. 3.3.



reference_17 Bass-clarinet arp Stereo.wav
reference_18 Bassoon arp,mp Stereo.wav
reference_19 Contra-bassoon arp,mp Stereo.wav
reference_20 Saxophone arp,mp Stereo.wav
reference_21 Trumpet arp,mp Stereo.wav
reference_22 Trombone arp,mp Stereo.wav
reference_23 Horn arp,mp Stereo.wav
reference_24 Tuba arp,mp Stereo.wav
reference_25 Harp arp,mp Stereo.wav
reference_26 Claves st,rhythm Stereo.wav
reference_27 Castanets st-rhythm Stereo.wav
reference_28 Side drum st,roll Stereo.wav
reference_29 Bass drum st Stereo.wav
reference_30 Kettle-drums st Stereo.wav
reference_31 Cymbal soft,hard stick Stereo.wav
reference_32 Triangles Stereo.wav
reference_33 Gong forte,piano Stereo.wav
reference_34 Tubular bells mp Stereo.wav
reference_35 Glockenspiel arp,mp Stereo.wav
reference_36 Xylophone st,mp Stereo.wav
reference_37 Vibraphone mp Stereo.wav
reference_38 Marimba st Stereo.wav
reference_39 Grand piano arp,st,mp Stereo.wav
reference_40 Harpsichord arp,mp Stereo.wav
reference_41 Celesta arp,st,mp Stereo.wav
reference_42AccordionmpStereo.wav
reference_43 Organ arp,mp Stereo.wav
reference_44 Soprano Stereo.wav
reference_45 Alto Stereo.wav
reference_46 Tenor Stereo.wav
reference_47 Bass Stereo.wav
reference_48 Quartet Stereo.wav
reference_49 Female speech English Mono.wav
reference_50 Male speech English Mono.wav
reference_51 Female speech French Mono.wav
reference_52 Male speech French Mono.wav
reference_53 Female speech German Mono.wav

Figure 3.3: Playlist

Option for selection of bitrate, packet loss rate are available but a maximum of four value vector can be selected for each.

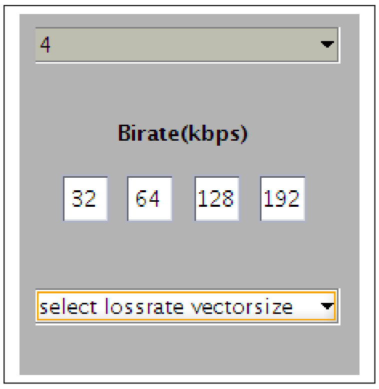


Figure 3.4: Bitrate and packet loss rate selection

The user can also select one or four of the allowed block size values

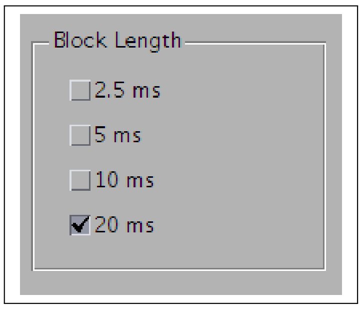
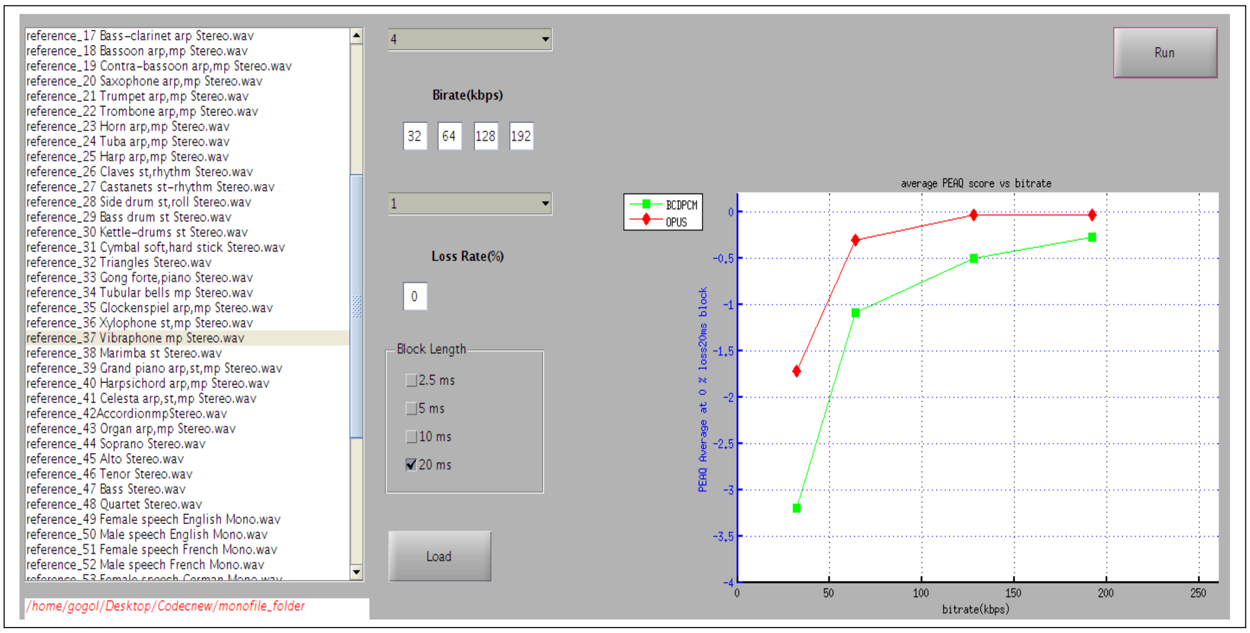


Figure 3.5: Block length Selection

At the end of a sample run the average ODG scores will be plotted in the the GUI plot area



reference_17 Bass–clarinet arp Stereo.wav
reference_18 Bassoon arp,mp Stereo.wav
reference_19 Contra–bassoon arp,mp Stereo.wav
reference_20 Saxophone arp,mp Stereo.wav
reference_21 Trumpet arp,mp Stereo.wav
reference_22 Trombone arp,mp Stereo.wav
reference_23 Horn arp,mp Stereo.wav
reference_24 Tuba arp,mp Stereo.wav
reference_25 Harp arp,mp Stereo.wav
reference_26 Claves st,rhythm Stereo.wav
reference_27 Castanets st–rhythm Stereo.wav
reference_28 Side drum st,roll Stereo.wav
reference_29 Bass drum st Stereo.wav
reference_30 Kettle–drums st Stereo.wav
reference_31 Cymbal soft,hard stick Stereo.wav
reference_32 Triangles Stereo.wav
reference_33 Gong forte,piano Stereo.wav
reference_34 Tubular bells mp Stereo.wav
reference_35 Glockenspiel arp,mp Stereo.wav
reference_36 Xylophone st,mp Stereo.wav
reference_37 Vibraphone mp Stereo.wav
reference_38 Marimba st Stereo.wav
reference_39 Grand piano arp,st,mp Stereo.wav
reference_40 Harpsichord arp,mp Stereo.wav
reference_41 Celesta arp,st,mp Stereo.wav
reference_42AccordionmpStereo.wav
reference_43 Organ arp,mp Stereo.wav
reference_44 Soprano Stereo.wav
reference_45 Alto Stereo.wav
reference_46 Tenor Stereo.wav
reference_47 Bass Stereo.wav
reference_48 Quartet Stereo.wav
reference_49 Female speech English Mono.wav
reference_50 Male speech English Mono.wav
reference_51 Female speech French Mono.wav
reference_52 Male speech French Mono.wav

/home/gogol/Desktop/Codecnew/monofile_folder

4

Birate(kbps)

32 64 128 192

1

Loss Rate(%)

0

Block Length

- ☐ 2.5 ms
- ☐ 5 ms
- ☐ 10 ms
- ☑ 20 ms

Figure 3.6: Interface

Apart from the user interface, the BCDPCM encoder and decoder scripts are also written in C and their respective matlab executable files are generated, for a faster execution of the codec.

Chapter 4

Complexity and Latency

4.1 Complexity

The computational complexity of a codec is a very important since it defines the power consumption of the device it is installed on. Portable audio equipment running on battery power requires that applications installed consume minimum computational resources.

In an audio codec bulk of the computational load is shared by the encoder. Particularly, wireless receiving devices need low decoder complexity for better battery performance. However in the context of AoIP, for bidirectional audio transmission, the distribution of computational load between the encoder and the decoder is not very important. Computational complexity of a codec is provided in terms of instructions per second or operations per second and depends on the programming language and type of processor.

Opus and BCDPCM are originally written in C and Matlab respectively. So calculation of complexity in instructions per second and comparison is challenging.

A theoretical approach has been taken to compute block computational complexity in terms of number of arithmetic operations by referring to the algorithms that define each component of both the codecs as well as by analyzing the very basic block diagrams of certain components.

Time complexity can also be calculated for BCDPCM from Matlab profiler to provide an idea on the distribution of computational load across the codec.

OPUS:

Different modules in OPUS perform different computations with certain modules contributing to bulk of the load.

Pre and post-filter:

The pre-filter and the post-filter are 5-tap comb filter tuned to filter widely spaced harmonics of highly periodic content in the audio signal. The transfer function of the pre-filter is given as [3],

$$H(z) = 1 - g[a_{tp,2}(z^{-T-2} + z^{-T+2}) + a_{tp,1}(z^{-T-1} + z^{-T+1}) + a_{tp,0}(z^{-T}), \tag{4.1}$$

where tp denotes tap sets, $a_{tp,i}$ are the coefficients and T is the pitch period

The complexity of the filter is linear with respect to samples and controllable by varying the number of taps.

Pre and post-emphasis filters:

They are equalizers used to reduce the effects of spectral leakage. It has a simple structure and low complexity of O(N) per frame, where N denotes number of samples per frames. The transfer function of the pre-emphasis filter is given as [3],

$$H(z) = 1 - 0.85z^{-1}. \tag{4.2}$$

Lapped Transform:

Direct implementation of the lapped transform and its inverse will result in both multiplications and additions of quadratic order . In Opus the MDCT is realized by N/4-FFT and the the number of operations are reduced by a factor of as much as 100 and the complexity is non linear and can be given as O($Nlog(N)$) per frame where N is the number of samples per frame.

Coarse and Fine Quantization

Coarse quantization involves an estimation of band energies from the previous bands as well as from calculation involving previous frames. It uses a fixed 6 dB resolution for all bands.

In fine quantization the quantizer resolution varies according to the bit allocation strategy.

The computational load of the overall scalar quantization procedure can be described by O(N_f) complexity as well where N_f denotes the number of frames.

Vector Quantization

The complexity of a pyramid vector quantizer depends on the dimension(D) of the normalized vector of transform coefficients and the number of pulses(K) representing the codevector. So the per band maximum complexity of a PVQ can be written as a function of DK per frame

BCDPCM:

For BCDPCM most of the computations are done in the adaptive prediction filter and the recalculation of normalization factors increase the recursive computation.

Prediction Filter

The robust prediction filter involves 3 multiplications and 2 additions in each stage of its prediction. Update of related coefficients also requires added calculations. However the algorithmic complexity remains a linear function of number of samples for the encoder and the decoder.

Quantizers

- Lloyd-Max quantizer is a non uniform quantizer with fixed codebooks based on number of quantization bits(b) where a codebook search is made through the 2^b codebook intervals resulting in in a maximum of 2^b search operations per sample and 1 division due to normalization. However the overall computation is of order O(n) for the encoder and the decoder where n denotes the number of samples.
- Normalization factors are quantized with uniform quantizers where the operation of deducing the quantizer step size according to number of allocated bits is done followed by division of input sample by the step size and a shift of the rounded quantized index within a predetermined range. The overall computation is also a function of number of samples, n, and is given in terms of O(n) in the encoder and the decoder.

Approximated algorithmic complexity values are provided for important components in Opus encoder and decoder in the table below. Computations imply the arithmatic operations like addition, multiplication, subtraction, and division. The values provided in the table are approximated formulations with respect to frame size, number of frames, and other variables.

Modules	**Computation**
Encoder	
Pre-emphasis Filter	$2NN_f$
Pre-Filter	$x_0(2tp+1)NN_f$
MDCT	$(2N\log_2(N)+2N)N_f$
PVQ	$(x_1DK+D)N_f$ per band
Coarse Quantizer	$(4x_2)N_f$ per band
Fine Quantizer	(x_3N_f) per band
Decoder	
Coarse-Fine Dequantizer	(x_4N_f) per band
PVQ Decoder	$(x_5DK+D)N_f$ per band
De-emphasis Filter	$2NN_f$
Post Filter	$(2tp+1)NN_f$
IMDCT	$(2N\log_2(N)+3N)N_f$

Table 4.1: Approximate computational complexity for Opus blocks

In Table 4.1, N_f denotes number of frames, tp denotes number of taps, N denotes frame length, D denotes number elements of a unit normalized band vector, K is the number of pulses required to encode D elements of the vector, x_i are functions or integral variables.

The valgrind tool suite has a profiling tool called callgrind that provides a summary of CPU instructions executed by important modules in the C code of either codecs. The results of the callgrind summary can be visualized in a tool called KCachegrind. In this visualization tool the percentage distribution of computational load, over the important parts of the Opus and BCDPCM codecs, can be identified from a callee map.

Bulk of the computation in Opus encoder is due to the pitch prediction, for calculation of pitch periods of the pre and post-filter, the vector quantization and the MDCT, while the inverse transform and the vector dequantization, in the decoder, perform the bulk of the total computation. Fig. 4.1 and Fig. 4.2 illustrates the load shared by important modules.

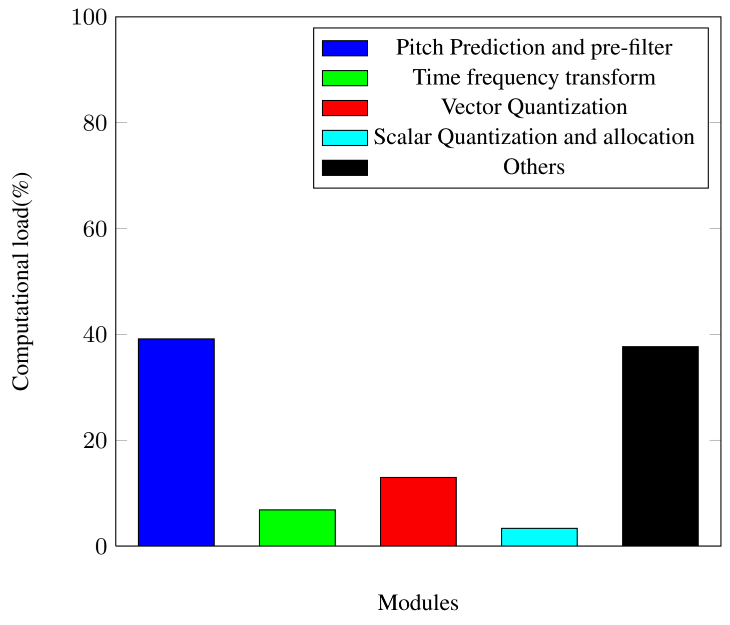


Figure 4.1: Distribution of computational load across Opus encoder

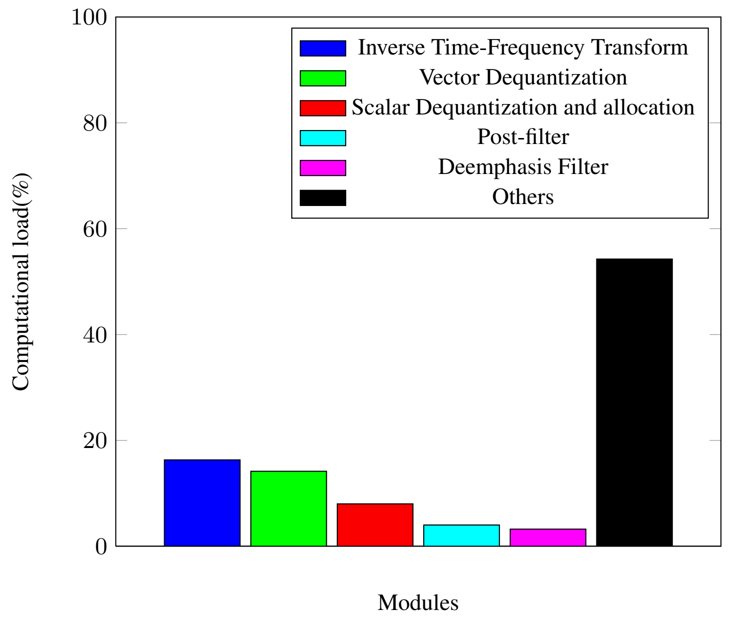


Figure 4.2: Distribution of computational load across Opus decoder

The "Others" module in Fig. 4.1 and Fig. 4.2 includes computations done in resampling, entropy coding, and other supporting functions. The input audio is a PCM data, sampled at 44.1 kHz, while the sampling rate in the Opus codec is 48 kHz. This results in resampling, and the amount of computation will be the same in both the encoder and the decoder. Since, the computational complexity of the decoder is lesser than the encoder, the computation due to resampling expressed as a fraction of total computation, is more, in case of the decoder than the encoder.

The formulations of computational cost with respect to parameters like number of samples, filter order, and other variables, are provided in Table 4.2.

Modules	**Computation**
Encoder	
Prediction Filter(with coefficient updation)	$40prn$
Max-Lloyd Quantizer	$(2^b + 1)rn$
Quantizer	$3r_{mod}n$
Decoder	
Prediction Filter(with coefficient updation)	$20pn$
Max-Lloyd Dequantizer	n
Dequantizer	n

Table 4.2: Approximate computational complexity for BCDPCM blocks

In Table 4.2, n denotes number of samples, r is a factor greater than or equal to 1 denoting number of recalculations, r_{mod} is a modified factor lesser than r, p is the prediction filter order, b is the number of allocated bits.

The direct and the inverse prediction filter is responsible for more than 75 $\%$ of the total computation in the BCDPCM codec as illustrated in Fig. 4.3 and 4.4.

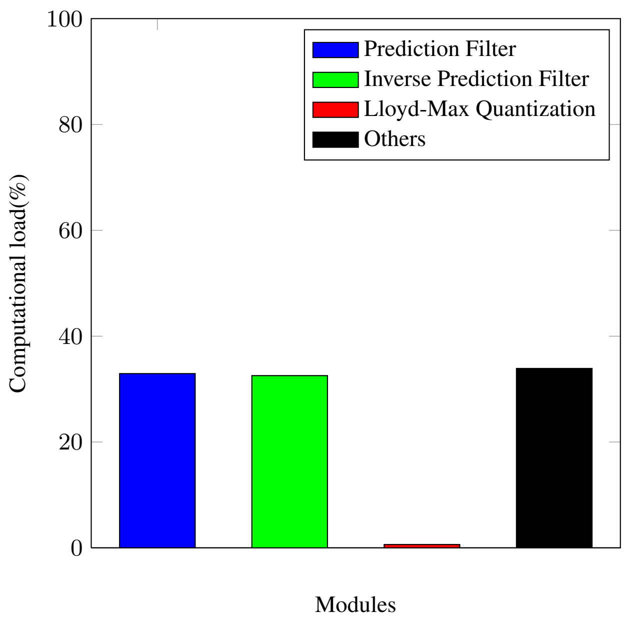


Figure 4.3: Distribution of computational load across BCDPCM encoder

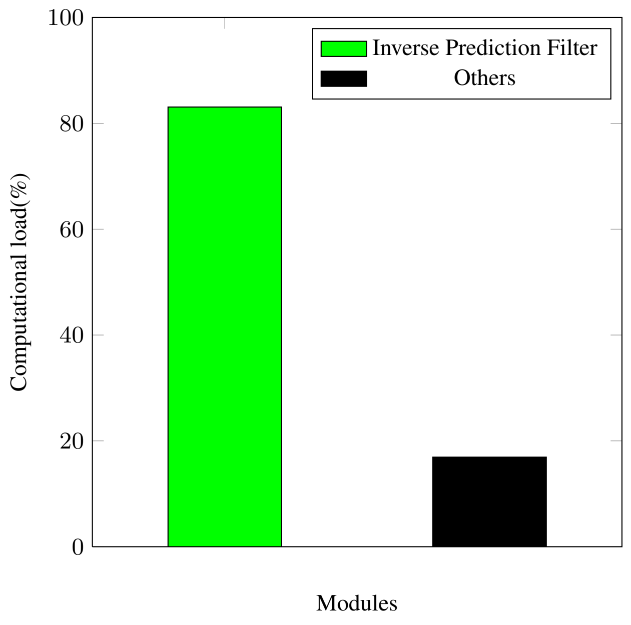


Figure 4.4: Distribution of computational load across BCDPCM decoder

The amount of computation done by the Opus encoder is almost twice the computation done by the decoder, while the complexity of the BCDPCM decoder is about 25 % of the complexity of the encoder. Fig. 4.5 and Fig. 4.6 display the load shared by the encoder and the decoder for Opus and BCDPCM respectively.

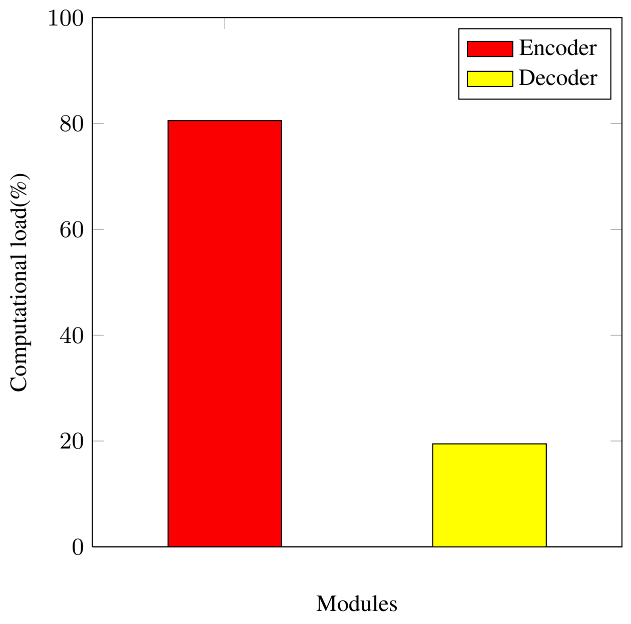


Figure 4.5: Distribution of computational load between BCDPCM encoder and decoder

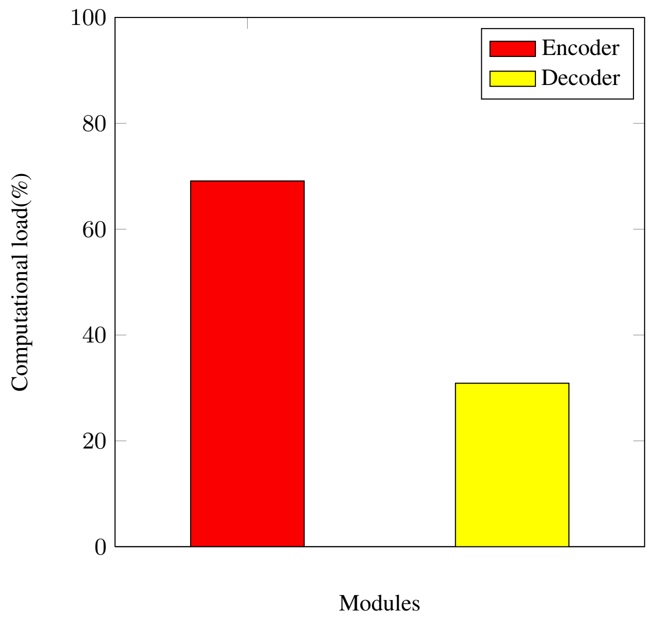


Figure 4.6: Distribution of computational load between Opus encoder and decoder

4.2 Latency

From audio signal generation to its reception there are various sources of delay that are introduced during its transmission process. Analog to digital converters (ADC), digital to analog converters (DAC), introduce very small latency around 1 ms. Routing the audio data requires time to cover the physical distance between source and target. Ideally an electromagnetic signal traversing every 300 km would introduce a delay of 1 ms, but transmission capability of physical medium, multitude of network devices and network capacity restrict the overall transmission rate rendering a non-deterministic delay. So transmission delay is practically dependent on the physical distance, the network conditions and the physical medium of transmission. However audio processing and audio compression tasks introduce appreciable amount of latency compared to the above.

If a audio codec introduces additional delay the total latency can become large, which is unsuitable for networked audio performance. An audio codec should introduce very low delay.

Audio codec delay depends on the following aspects

- Filters: Depending on the filter and its length some delay might be introduced in the audio stream.

- Framesize: For blocked operations the size of the block or frame introduces the delay. So for low delay operation short frame sizes are used.

- Look ahead: An additional delay is introduced if information from a neighbouring frame is required for computation.

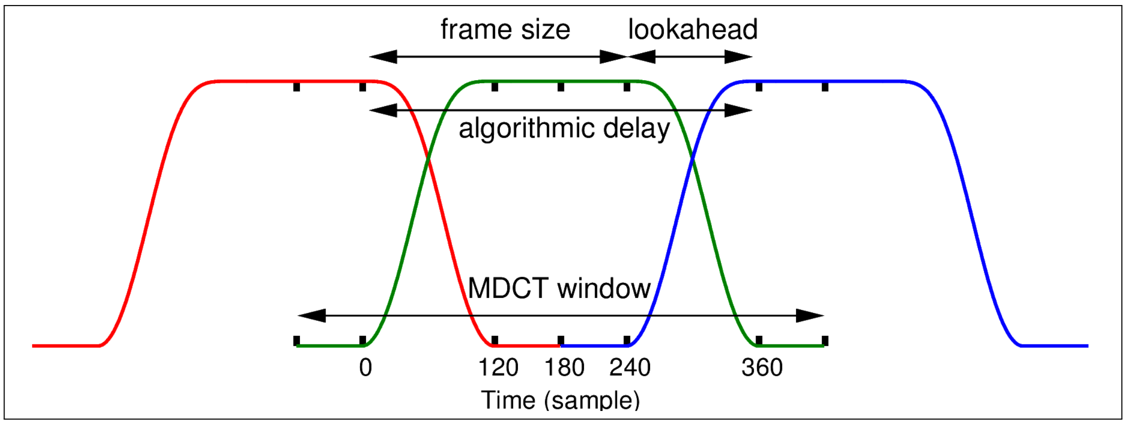


Figure 4.7: Overlapping windowed frame [3]

For Opus codec the algorithmic delay introduced in celt mode is equal to the framesize in ms as shown in Fig 4.7. An additional delay of 2.5 ms is introduced due to look ahead. The results are tabulated in Table 4.3

Since BCDPCM does sample wise processing, delay is expected to be zero. However BCDPCM introduces negligible delay and with companding block length as low as 16 the algorithmic delay is equivalent to 16 samples.

Framesize(samples)	**Delay (ms)**	**Look ahead (ms)**	**Total Delay (ms)**
960	20	2.5	22.5
480	10	2.5	12.5
240	5	2.5	7.5
120	2.5	2.5	5

Table 4.3: Algorithmic Delay in Opus

Maximum algorithmic delay in BCDPCM for a companding block size of 16 samples can be given as $\frac{16}{44100}$ seconds or .36 ms which is very low.

The process of resampling involves computation done by digital filters. These filters introduce a small amount of lag in the output and the delay depends on the type of the filter, its design, and its attributes like filter length and resampling factor. Resampling is done in both the codecs and hence a very small delay will be added to the existing delay.

Chapter 5

Results

This chapter displays the graphical result set of under different run configurations for both of the codecs. Test simulations are done for lossy and lossless transmission of packets from the encoder to the decoder. The simulation is done on the audio files from the SQAM dataset.

The results include magnitude plots in time and frequency domains, and the quality plots measured by PEAQ for a selected set of audio files. The ODG scores of each individual run and the average ODG scores are saved in a data structure, during the test procedure, and are plotted against the input parameters for each audio track.

5.1 Simulation without packet loss

At lower bitrates, the quality of output from BCDPCM is relatively low compared to that of OPUS when there is no packet loss.

Due to resampling, the effective sampling rate is reduced in BCDPCM at lower bitrates, and thus the effective Nyquist frequency is also reduced. The entire bandwidth of the audio signal cannot be reconstructed properly.

Using higher bitrate the quality of BCDPCM audio output improves, and for some audio signals, the quality is deemed better than Opus.

For smaller frame sizes Opus suffers the effects of leakage. Hence for shorter frames the quality downgrades for the input audio files. Combined with packet loss ODG scores for 2.5 and 5 millisecond frames are very low.

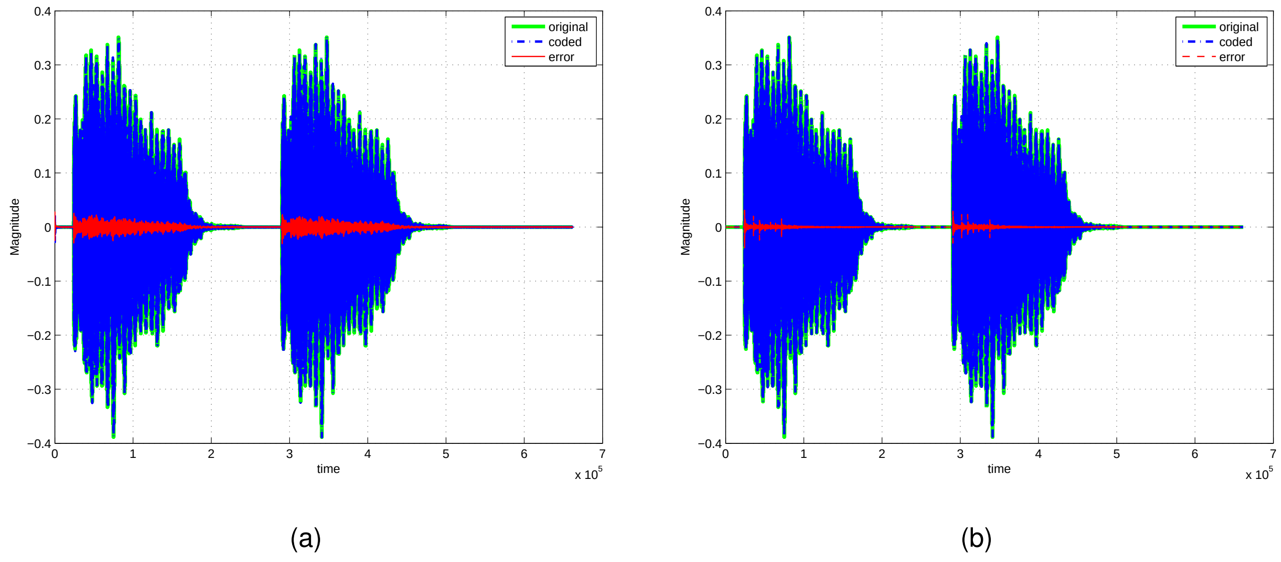


Figure 5.1: Magnitude versus time plot of Vibraphone for (a) Opus (b) BCDPCM at 64 kbps for 10 ms frame

Fig. 5.1 provides the magnitude plot over time. BCDPCM output closely resembles the reference signal while apparently the magnitude of error in Opus is greater.

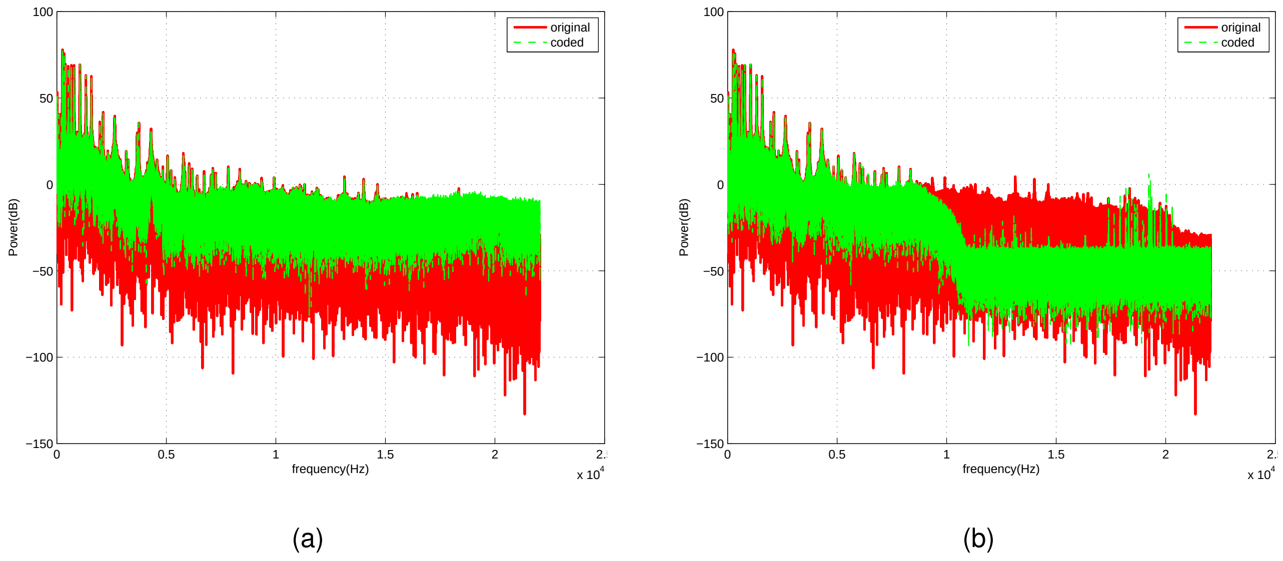


Figure 5.2: Power spectrum of Vibraphone for (a) Opus (b) BCDPCM at 64 kbps for 10 ms frame

According to Eq. (3.1), the effective sampling rate for a bitrate of 64 kbps and quantization of 3 bits per sample of residual and 6 bits per sample of normalization factor, is nearly $\frac{64}{3.375}$ or 19 kHz. So the input audio signal becomes band limited and cannot encode audio content beyond 9.5 kHz properly as illustrated in Fig. 5.2(b). As already mentioned in chapter 2, that, the bandwidth of the signal is an important aspect for PEAQ grades, BCDPCM suffers at lower bitrates.

Individual and average ODG scores are calculated for audio test signals in the SQAM database at bitrates of 64, 128, 192, and 256 kbps.

The ODG scores for Vibraphone and Cymbals respectively at different frame sizes are displyed in Fig. 5.3 and 5.4. The quality of audio output for the Opus and the BCDPCM codecs improve with increasing bitrate.

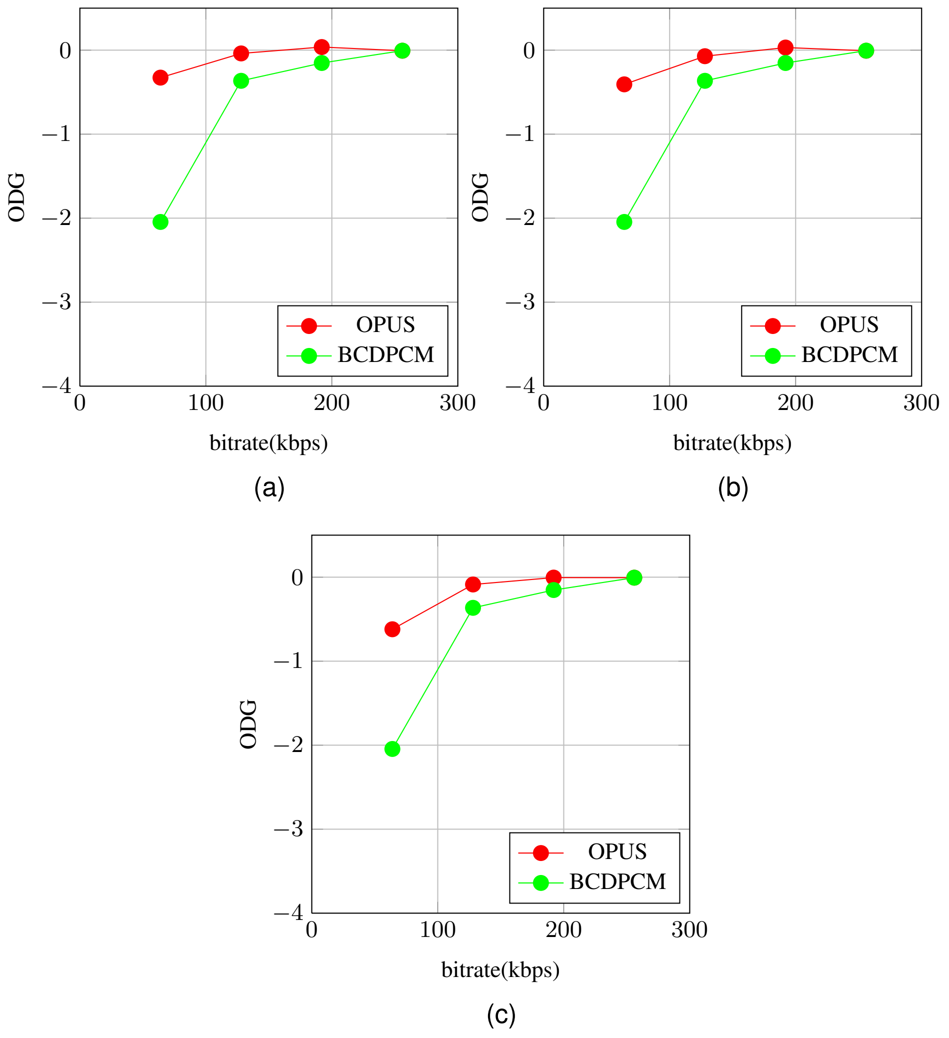


Figure 5.3: PEAQ scores for Vibraphone with (a) 20 ms (b) 10 ms and (c) 5 ms frame sizes

Fig. 5.5 provides the average ODG scores of all the audio files in SQAM dataset under different values of control parameters. The audio quality between Opus and BCDPCM are very close for high bitrate. Since there is no packet loss the frame size will not affect the BCDPCM results. This is evident in Fig. 5.3 , 5.4, and 5.5. However a closer look at the plots reveal very small degradation in output audio quality for Opus with decreasing frame size.

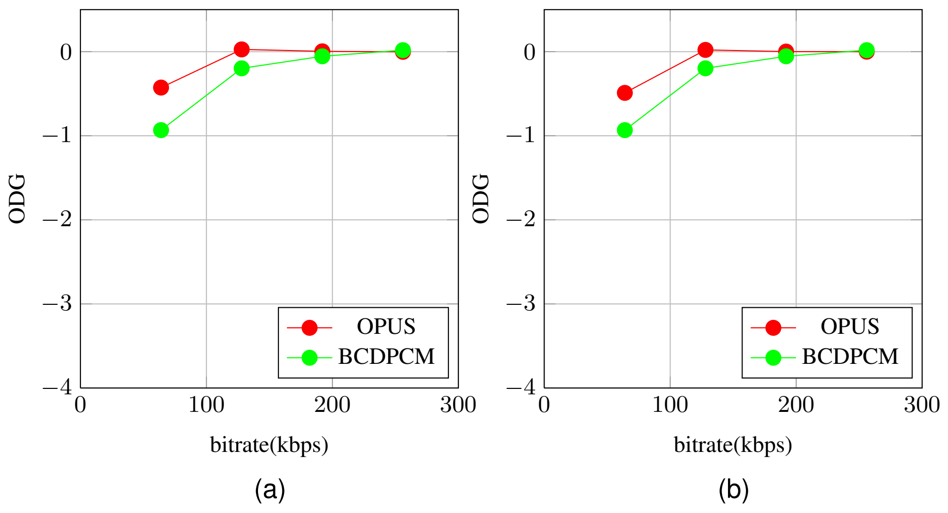


Figure 5.4: PEAQ scores for Cymbals with (a) 20 ms (b) 10 ms frame sizes

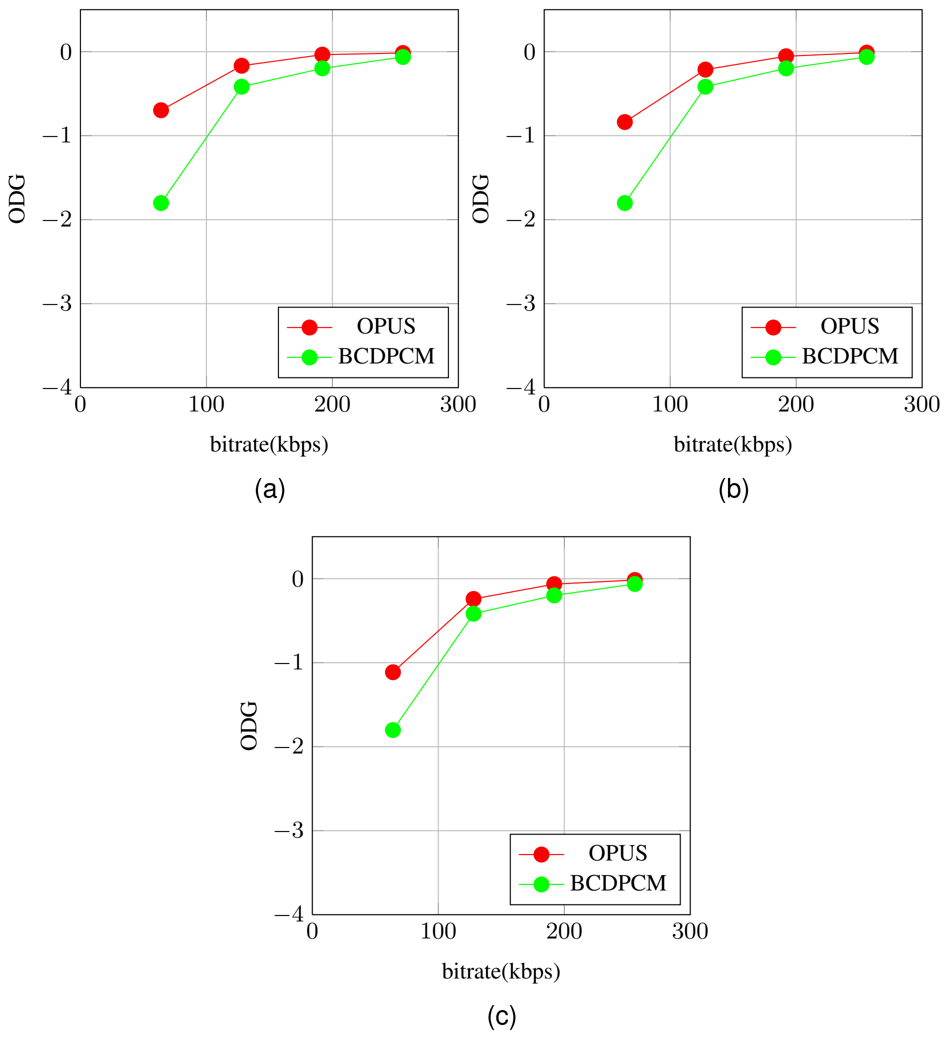


Figure 5.5: Average PEAQ scores for (a) 20 ms (b) 10 ms (c) 5 ms frame sizes

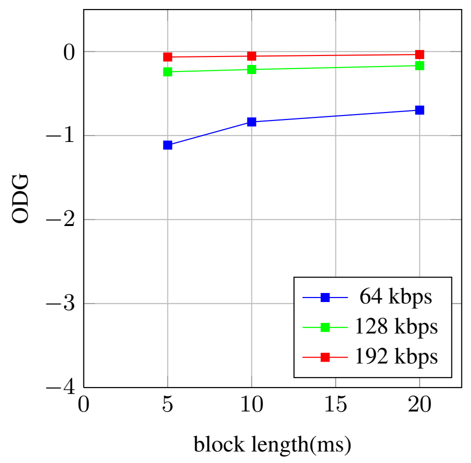


Figure 5.6: Average PEAQ scores for different frame sizes for Opus

Frame size affects the quality of audio encoded and decoded by Opus. The spectral leakage, due to small sized frames, causes quantization errors particularly if the number of bits available for quantization is small. These errors reduce the audio quality and the ODG scores are low. With increment of framesize and bitrate, the audio quality improves and at higher bitrates, the increase of ODG scores is very small, compared to the lower bitrates, as illustrated in Fig. 5.6.

The trend of improvement, in quality of audio, encoded and decoded by BCDPCM, is much steeper than in Opus since at higher bitrates, the signal does not become band limited for BCDPCM unlike, at lower bitrates.

Fig. 5.7 illustrates the ODG scores for a set of tracks from the SQAM dataset for no packet loss at 128 kbps for a 20 millisecond blocksize. The average ODG score for the audio tracks encoded and decoded by Opus is about -0.07 while the average score is about -0.41.

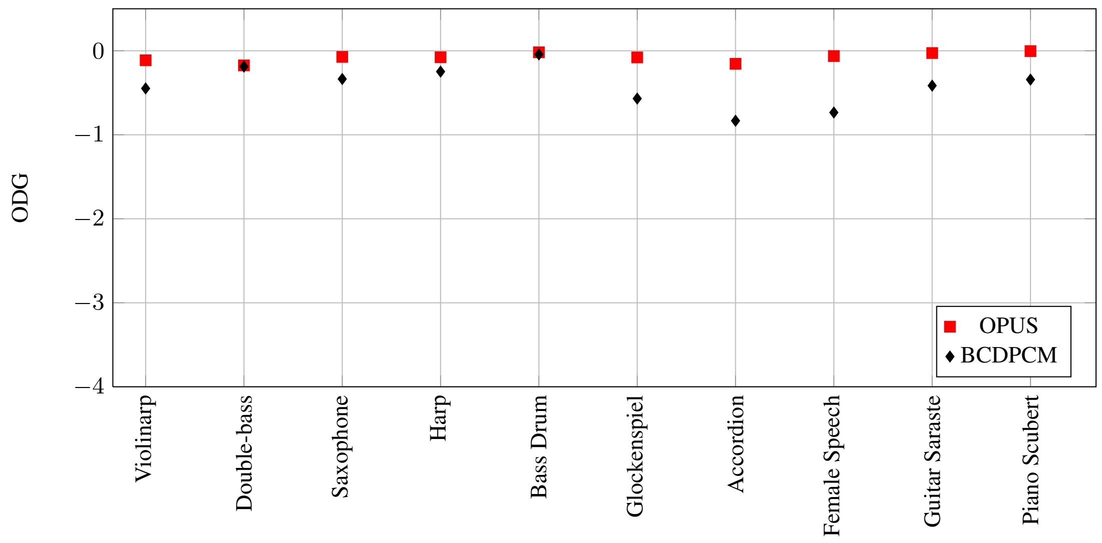


Figure 5.7: PEAQ scores for selected SQAM files at 128 kbps for 20 ms blocksize

Although both of these codec produce good quality of output audio, Opus produces a better result. This gap reduces for higher bitrates. In the next section the quality results under packet loss are provided. The quality of the encoded and decoded audio also depends on the type of original audio track. The information content in an audio or its entropy, the presence of any source noise or any dominating trait in the audio like tonality, transience or silence, characterizes the audio signal. Opus codec employs measures to deal with different types of audio signal or the audio signal content. Under the the same parametric setup, the ODG scores of different tracks, encoded and decoded by Opus, do not have a large difference.

For BCDPCM the quality of decoded audio depends on the accuracy of the prediction by the GAL algorithm in the encoder. For some audio tracks, the prediction filter is able to closely follow the input audio track and produce a residual signal with very low variance, better than some other audio tracks. The quality of the decoded Bass Drum and the Double Base track is better compared to the rest. The measure of the variance of their residual signal is very low. The Shannon entropy values of the residual signal output, from the BCDPCM encoder, are calculated for some audio tracks with the wentropy function in Matlab[8] and the measured values are illustrated in Fig. 5.8.

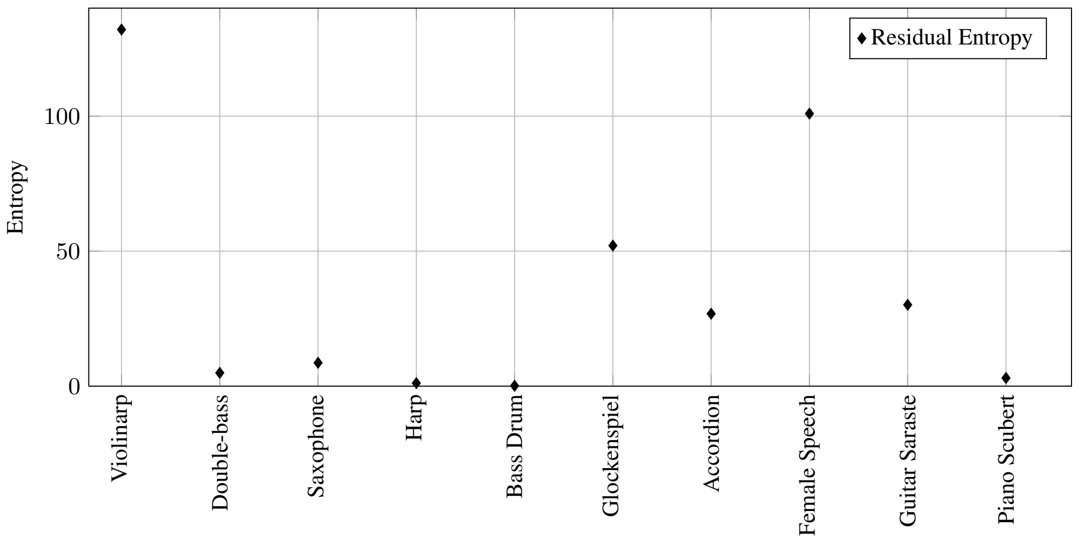


Figure 5.8: Measure of entropy of residual signal for selected SQAM audio tracks at 128 kbps for 20 ms blocksize with the wentropy function

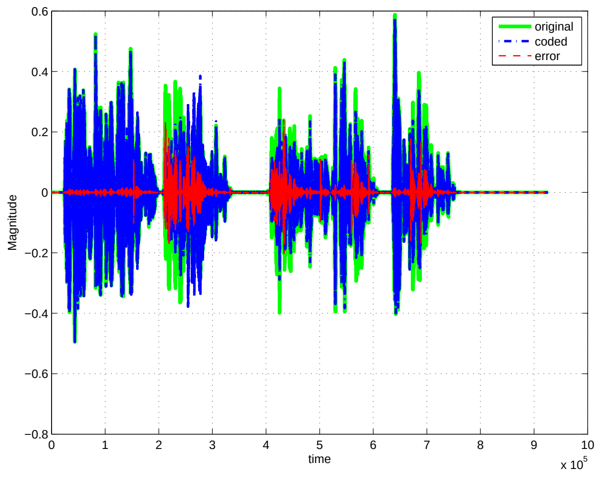
In Fig. 5.8, it can be seen that the entropy value of the residual is very low for both bass drum and double bass. Also, the reduction of this value from the original signal to the residual signal is more for bass drum and double bass compared to others.

For the cymbals track there are long periods of silence between each attack, which can be predicted very accurately and the measure of variance of the encoded residual is low. So, at higher bitrates the quality of audio output from BCDPCM is very good. BCDPCM codec employs a sample-wise encoding and decoding process. So, unlike Opus there is no concern about time resolution adjustments for percussive tracks. As illustrated in Fig. 5.4 the ODG scores for Opus and BCDPCM are very close for higher bitrates.

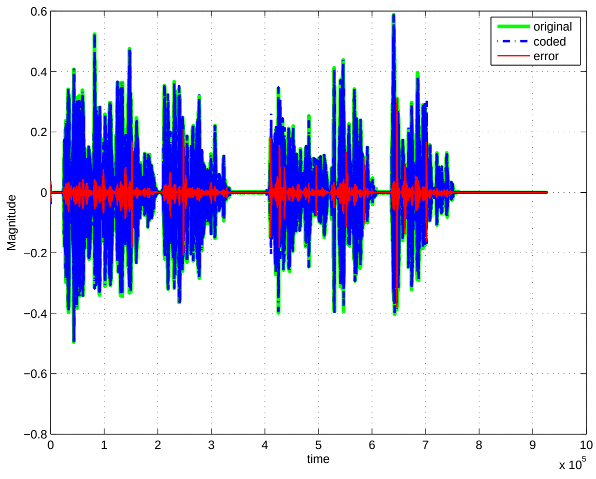
5.2 Simulation with packet loss

BCDPCM is heavily affected by packet losses. The Modified GAL algorithm attenuates the adverse effects of deviation from the desired output signal envelope but is effective as long as the packet size of the lost packet or the number of lost packets is reasonably small. It provides appreciable remedy against bit errors but against packet loss or block errors it ceases to be influential.Indeed at higher bitrates, where number of packets lost or size of the packet is more compared to that in lower values, the PEAQ scores decrease rapidly.

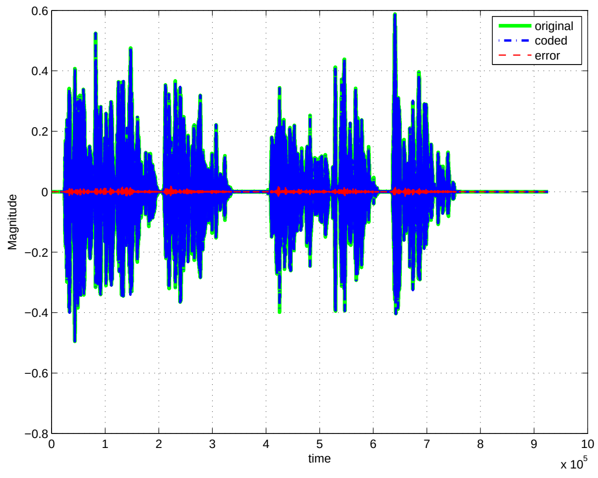
In this context BCDPCM packets containing equal number of sample, as in Opus, along with overhead data, is transmitted to the decoder and lost randomly according to percentage of loss.



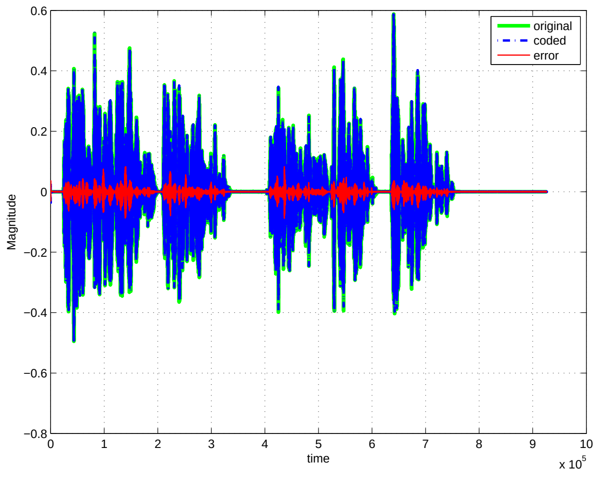
(a)



(b)



(c)



(d)

Figure 5.9: Magnitude versus frequency plot of German male speech for (a) BCDPCM (b) Opus at 128 kbps for 20 ms framesize and 2% loss and (c) BCDPCM (d) Opus at 128 kbps for 20 ms framesize and no loss

The magnitude plot in Fig. 5.9 indicate that the magnitude of error in the male speech signal encoded and decoded by BCDPCM is more compared to Opus at 2% packet loss. The output audio of BCDPCM codec has suffered more due to packet loss than the audio output from Opus codec. Since the BCDPCM codec employs a prediction algorithm that estimates current value of a sample from the past values, an error in sample value due to data loss results in erroneous prediction of the current value. This error in the estimation is carried over to the future samples as well.This issue is referred to as mistracking. The leakage factors discussed in Chapter 2, reduce the effect of mistracking but, for considerable size of lost data, the error spreads over a large number of samples, beyond the size of the lost packet.

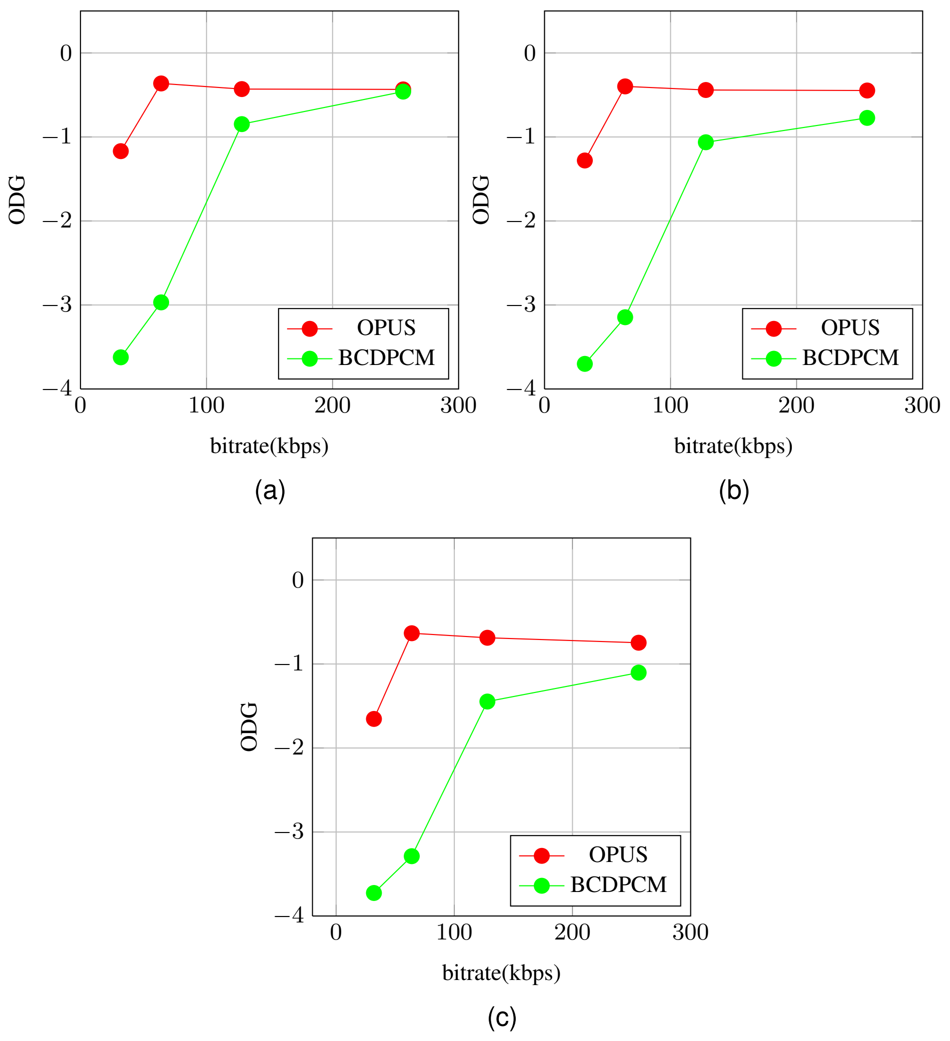


Figure 5.10: PEAQ scores for German male speech with 20 ms framesize at (a) 0.5% loss (b) 1% loss (c) 2% loss

Fig. 5.10 illustrates the ODG scores of the decoded German male speech at different values of packet loss percentage. The audio quality for both Opus and BCDPCM degrades with bitrate. At 128 kbps, the ODG score decreases from 0.01, when there is no packet loss, to -0.85 ,for 2 percent packet loss,in case of Opus. For BCDPCM the same reduction occurs from -0.6 to around -2.5. In the Fig. 5.10 it can be seen that audio quality degrades more for BCDPCM compared to Opus, with increasing percentage of packet loss (The gap between the red and the green dots, increases with packet loss).

At higher bitrates, the number of lost packet is very large for BCDPCM due to increment of number of samples via resampling. For the BCDPCM codec, higher bitrates (> 176 kbps) result in upsampling which involves inclusion of new sample values. This can be a source of error in the first place.

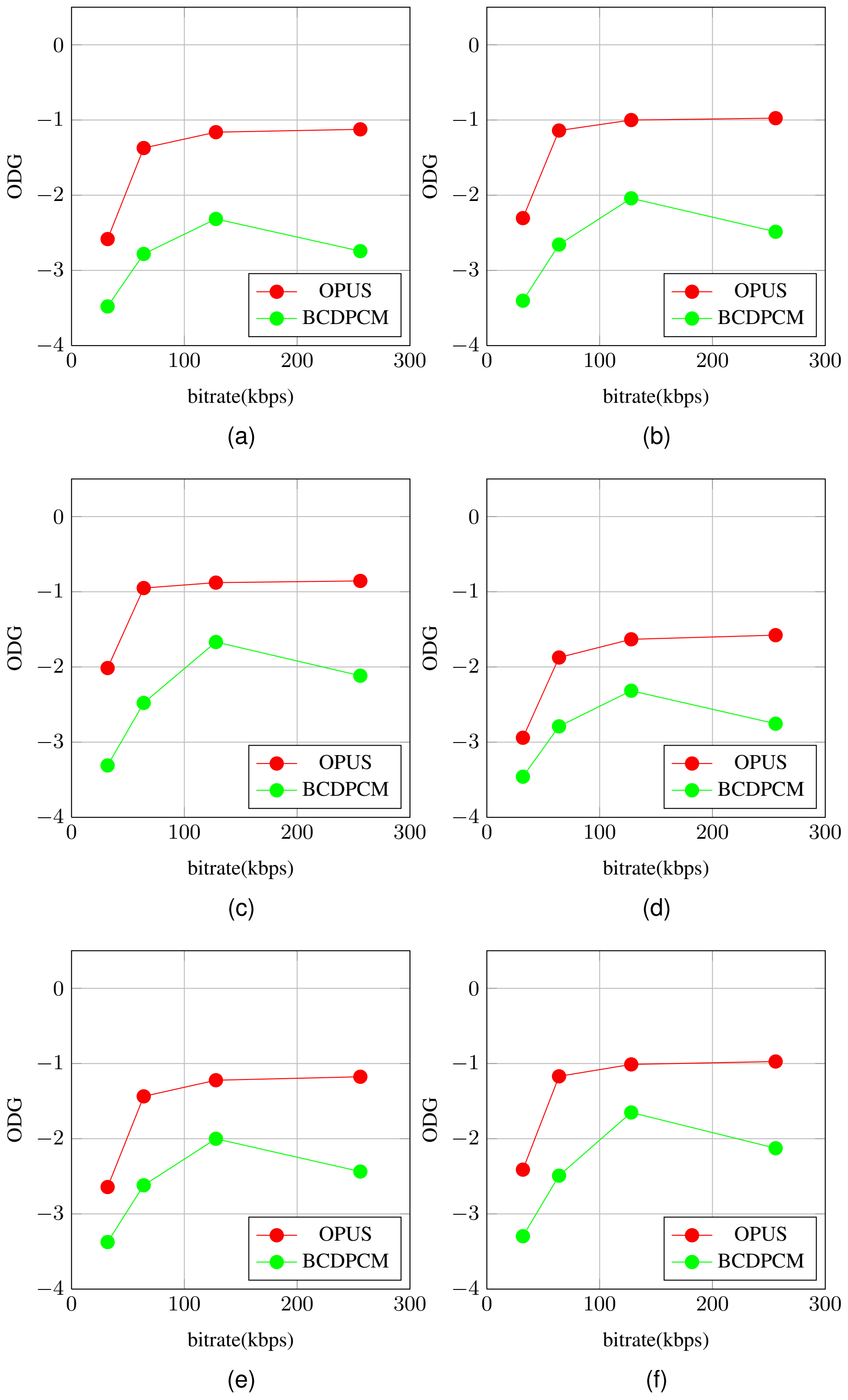


Figure 5.11: Average PEAQ scores with 20 ms frames at (a) 2% loss (b) 1% loss (c) 0.5% loss and 10 ms frames at (d) 2% loss (e) 1% loss (f) 0.5% loss

Upsampling results in an increase of the total number of samples of the audio track to be processed. If the number of samples per packet, transmitted to the decoder, is constant then for a given percentage of packet loss, the number of lost packets is more than in the case of lower bitrates. Depending on the audio content this can result in further degradation of quality. With increase in the number of lost packet the probability of losing perceptually important audio content increases. So for BCDPCM, at very high bitrate many audio tracks will suffer a drop in quality compared to lower bitrates. Opus do not suffer from the above issues. So there is a drop in ODG score for many audio tracks at higher bitrates, as illustrated in Fig. 5.11.

Packet loss simulation is also done in a different way. The total number of packets or frames, N_f, processed by Opus and the indices of the packets lost in Opus or the loss distribution, are recorded. The total number of samples to be transmitted to the decoder of the BCDPCM codec are divided into N_f packets and the same loss distribution is followed in case of BCDPCM. However for different bitrates, the packetsize will vary in BCDPCM. For higher bitrates, upsampling will result in more number of sample per packet, which is going to invoke extended mistracking.

The shorter and scarcer the duration of the error, the faster the signal could be prevented from being mistracked.

Fig. 5.12 highlights the loss of data and mistracking occuring in Opus and BCDPCM respectively due to packet loss of 2 percent, at 192 kbps, in a speech sample.

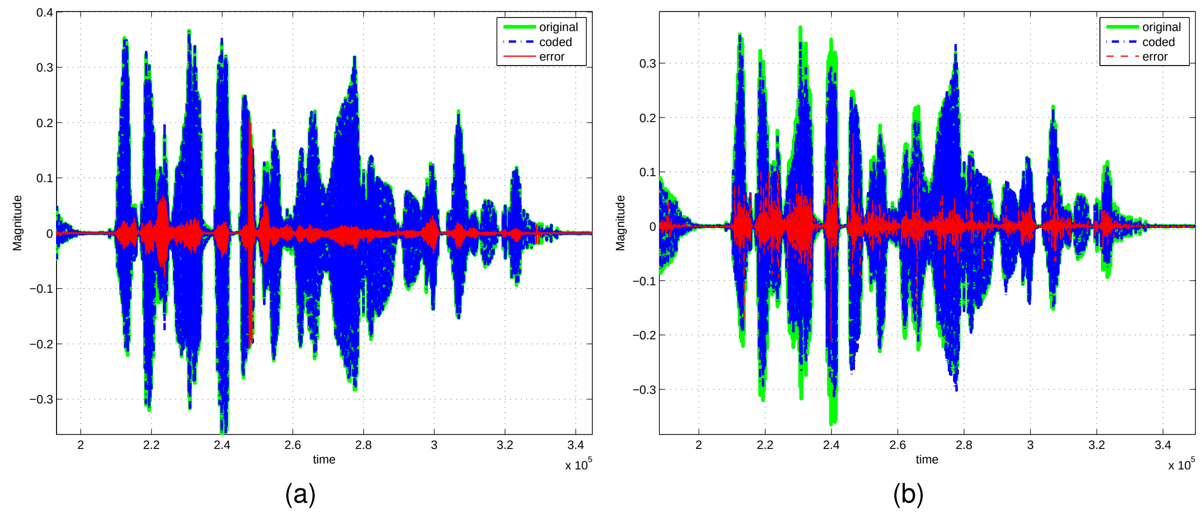


Figure 5.12: (a) Packet loss in Opus (b) Mistracking in BCDPCM

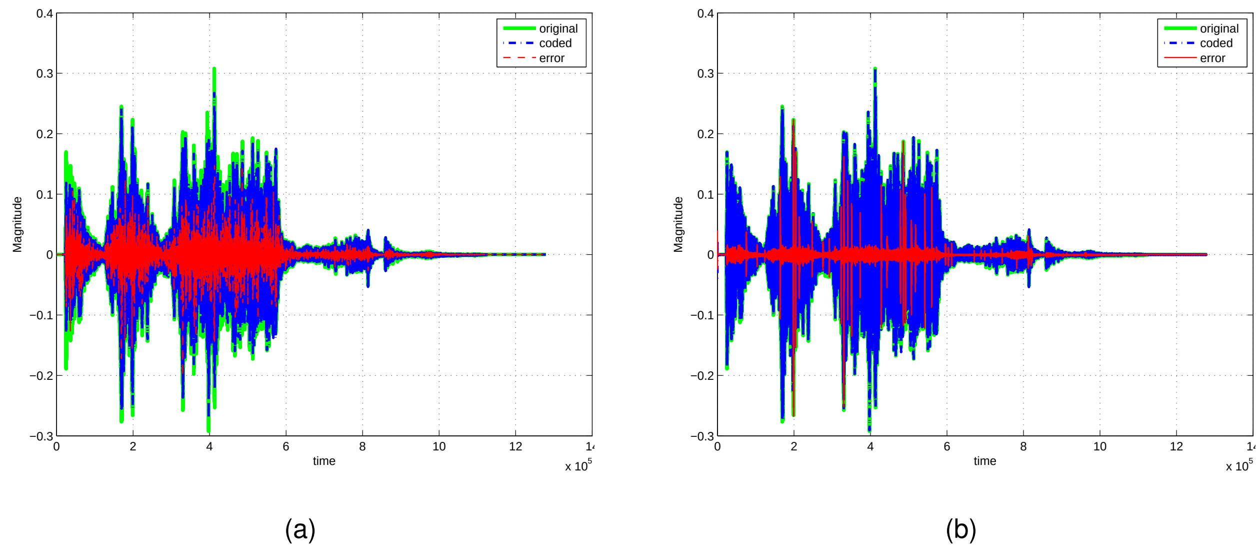


Figure 5.13: Magnitude versus time plot of (a) BCDPCM (b) Opus output for Violin Ravel at 192 kbps and 2% loss for 10 ms block

In case of Opus the packet loss does not greatly affect the next samples. In Fig. 5.12, the sharp red spike at an index of 2.5 on the time axis, denotes the packet lost in Opus. In case of BCDPCM, it can be seen that, due to packet loss and transmission of incorrect residuals to the decoder, the reconstructed signal deviates from the original and this error is spread accross the neighbouring samples, unlike Opus.

Fig. 5.13 shows the magnitude versus time plots for Violin Ravel. With 2% packet loss the error signal in BCDPCM is greater compared to Opus due to extensive mistracking.

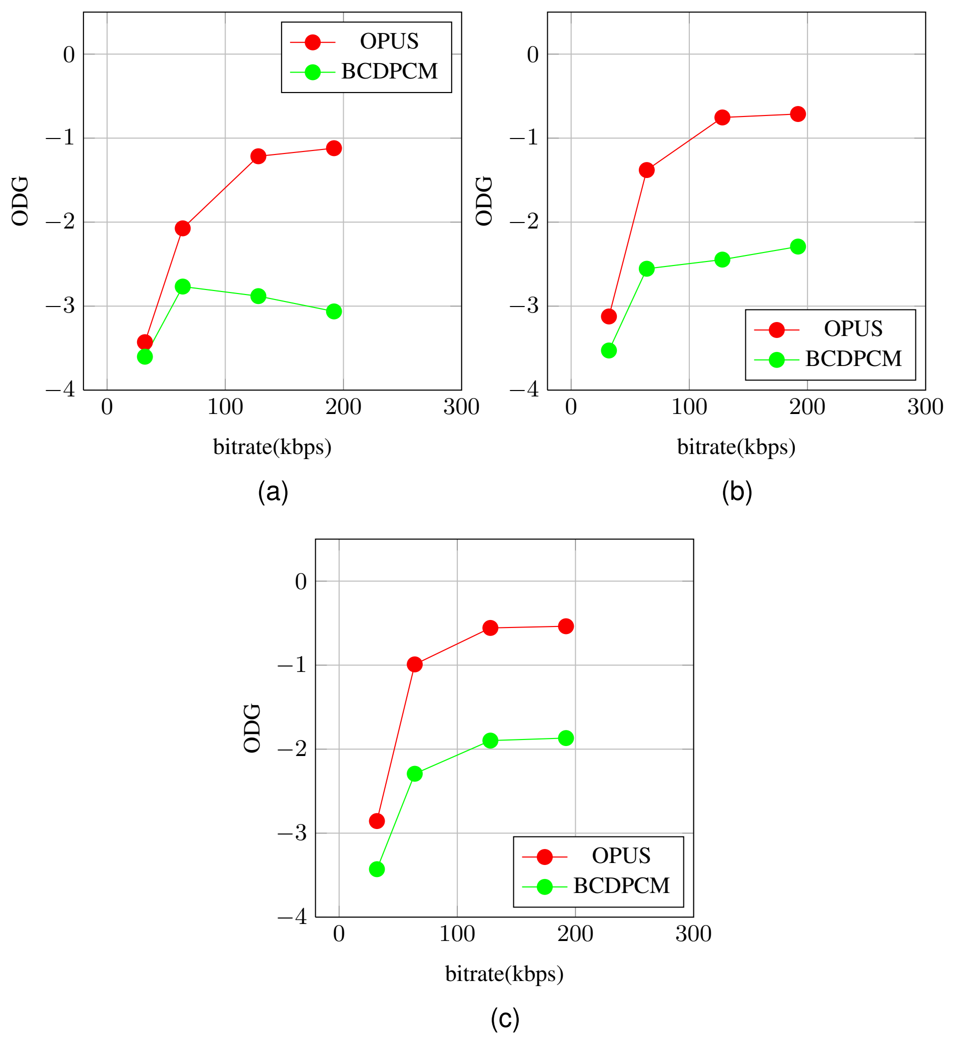


Figure 5.14: PEAQ scores for Violin Ravel with 5 ms framesize at (a) 2% loss (b) 1% loss (c) 0.5% loss

The ODG scores of the decoded Violin Ravel audio at different values of packet loss percentage is illustrated in Fig. 5.14. The audio quality improves, for both Opus and BCDPCM, when the percentage of packet loss decreases.

The average ODG score over all the audio tracks in SQAM dataset is illustrated in Fig. 5.15. The audio quality drops with increase of packet loss percentage. At higher bitrates, the size of the lost packet is very large for BCDPCM and depending on the audio content this can result in further degradation of quality. Leaky GAL requires the amount of lost data to be small and discontinuous, to avoid extensive mistracking. So for BCDPCM, at very high bitrate many audio tracks will suffer

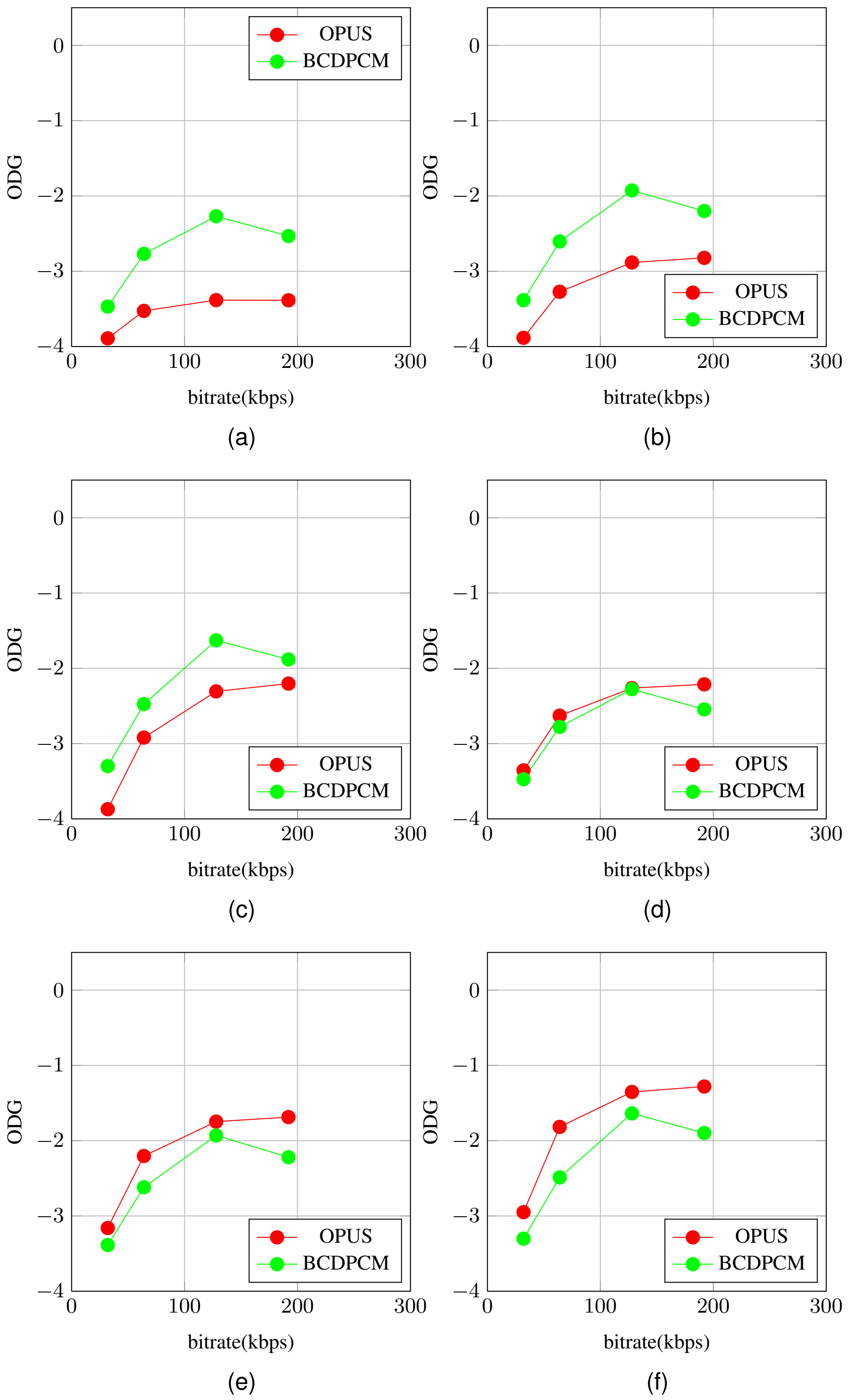


Figure 5.15: Average PEAQ scores with 2.5 ms frames at (a) 2% loss (b) 1% loss (c) 0.5% loss and 5 ms frames at (d) 2% loss (e) 1% loss (f) 0.5% loss

a drop in quality, compared to lower bitrates. A keen look will reveal very small improvement in quality in BCDPCM with decrease of packet size or frame size.

However for Opus quality suffers with reduced packet or frame size.

In the Fig. 5.15 it can be seen that for 5 millisecond frames the average audio quality for Opus and BCDPCM are very close and as already discussed for small frame size of 2.5 millisecond results from Opus are poorer than BCDPCM.

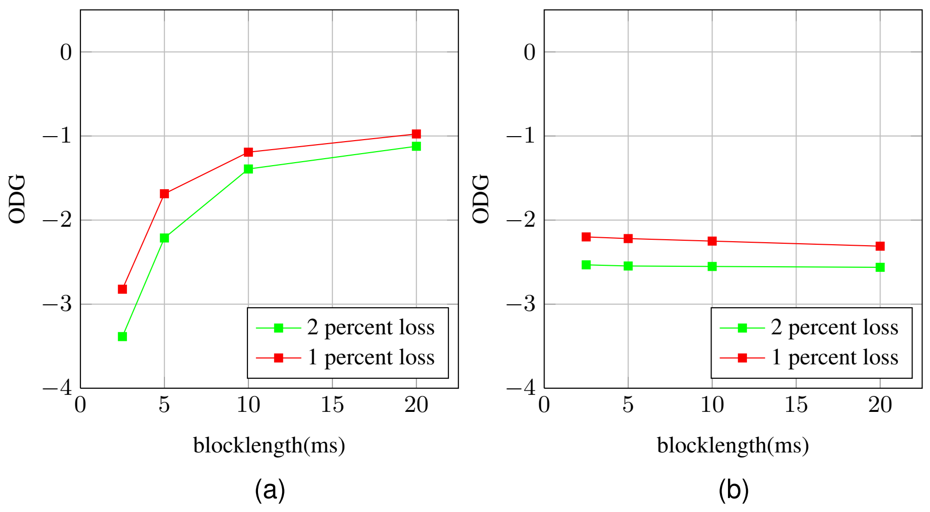


Figure 5.16: Average PEAQ scores at 192 kbps for (a) Opus (b) BCDPCM for different frame sizes and packet loss percentage

In the event of a packet loss , the audio quality in Opus degrades further if small frame sizes are used, due to spectral leakage. However, in the case of BCDPCM, smaller sized packets ensure that the adverse effects of mistracking is less, compared to the case of larger packets as illustrated in Fig. 5.16.

The average ODG scores of audio output from the Opus codec, becomes better with increasing frame size while there is a small decrease in ODG scores in case of the BCDPCM codec.

From the results it is clear that Opus requires larger frames to avoid adverse effects of leakage particularly at lower bitrates. This would however result in an increase in algorithmic delay. While BCDPCM do not suffer from the above issue, the codec requires to be more robust to packet loss which would result in an increase in computation.

Chapter 6

Conclusion

This chapter provides a summary of the previous chapters along with general conclusions and scope for improvements and further work.

A short introduction about the purpose of this work is provided in Chapter 1. Chapter 2 provides the fundamentals about the Opus codec and the BCDPCM codec and their operation. An introduction to the audio quality evaluation tool is also provided in this chapter. Chapter 3 explains the setup of the test procedure. A theoretical study of latency and computational complexity associated with the codecs are done in chapter 4 and the corresponding results are provided. Chapter 5 provides the test results of the audio quality evaluation and related explanations.

On the evidence of the simulation results it can be concluded that at lower bitrates (< 100 Kbps) BCDPCM and Opus results differ, by a good extent. Under lower data rates quality of audio in Opus is better and thus achieves same compression, but with lower degradation of audio quality than BCDPCM. However this gap reduces with increasing bitrate. In terms of latency BCDPCM is better than Opus as it introduces almost zero algorithmic delay. Indeed, there is a deterioration in quality for Opus with decreasing latency. Since delay introduced is very important, in multipath audio communication, BCDPCM has an advantage. Opus is a moderately complex codec but offers the user to control the complexity through reduction in computation in important functions including pitch prediction, band energy prediction, bit allocation techniques. However under lowest complexity setting its quality will degrade. Algorithmic complexity in BCDPCM is high and has a direct non linear relation with bitrate. So at lower bitrates, the overall complexity reduces at the cost of quality. The biggest drawback in BCDPCM is its robustness or the lack thereof, to block errors. As already mentioned in the previous chapters the current algorithm provides low security against loss of larger packets. Opus has a forward error correction strategy that involves re-transmission of selected packets depending on the packet loss percentage and frame sizes thus increasing the overhead data as well as the algorithmic delay. However for low latency applications FEC is not suitable. For low delay music coding, packet loss concealment strategy is present, based on pitch prediction and subsequent interpolation of data from previously decoded packets, for a lost packet. This will amount to an increase in computation. The general attributes, which govern the decision of selecting a codec for diverse applications, are provided below.

- **Quality** : When there is no loss of packets, the average quality of audio, encoded and decoded by Opus at lower bitrates, is moderately good with and ODG score around -1. At higher bitrates, the ODG scores are in between 0 and -0.2. For BCDPCM, the audio quality is poor at lower bitrates and the average ODG score is around -2, while the score improves, to around -0.5, for higher bitrates. Thus, better quality of audio is achieved at the cost of lesser data compression. When there is a packet loss, the audio quality degrades for both the codecs.

- **Latency** : The Opus codec introduces an algorithmic delay of 5 ms to 20 ms while BCDPCM introduces negligible delay. As mentioned previously, traditional audio codecs like Mp3, Vorbis or AAC introduce moderate to high algorithmic delay making them unsuitable for AoIP applications. Thus BCDPCM and Opus can be selected ahead of them, in terms of latency.

- **Complexity** : Computational complexity of the Opus codec is moderately high but it can be controlled and reduced. The pitch prediction, time frequency transform, and the vector quantization contribute to bulk of the computation. The encoder performs, almost twice the amount of computation, compared to the decoder. The BCDPCM codec has high computational complexity and is dependent on the audio data itself. The prediction filter is responsible for bulk of the computation in this codec. The amount of computation done by the encoder is almost four times the computation performed by the decoder.

- **Robustness to packet loss** : Opus codec has a packet loss concealment strategy in place, to counteract the effects of packet loss on audio quality. Even without the concealment strategy the quality of decoded audio do not suffer as much as it does in case of the BCDPCM codec. The prediction filter estimates the current sample based on the prior samples and this stateful or non-memoryless property of the filter results in distribution of any error into the future samples. Currently there is no implementation of any efficient concealment strategy against packet loss. As a result, the quality of audio suffers due to packet loss.

- **Transmission overhead** : Transmission overhead includes the side information transmitted with the compressed audio data. Opus has a transmission overhead of 0.5-12 % for bitrates higher than 32 kbps. The amount of overhead, expressed in percentage, decreases with increasing bitrate and framesize. The BCDPCM codec has an overhead of 9.8 to 12.5 %

Clearly, resampling in BCDPCM contributes to the reduction in quality of audio and the codec is not robust to packet loss. These issues need to be addressed for improvement in quality of decoded audio. Provided BCDPCM introduces negligible latency, the codec can be used in AoIP at moderately high bitrates when there is almost no packet loss. In Opus a delay as low as 5 ms is achieved due to very small frame sizes but the quality is well reduced. The computaional complexity in Opus is comparatively low and hence can be used in both AoIP applications as well as applications involving low power devices.

Writing the algorithm of BCDPCM coder and decoder in C in a very optimized way, and running the codecs in a dedicated processing environment would yield a proper computation count and comparison of complexity. During simulation of a random packet loss for a given percentage, it is better to increase the number of runs for the same setting and take an average of the result. Subjective testing of audio quality or other objective testing procedures can be implemented, to achieve a diverse reference about the decoded audio quality. In terms of improvements, a efficient strategy for packet loss concealment for BCDPCM can be devised. Improving the robustness to block errors, offer scope in terms of research. The method of recalculation of normalization factors and the prediction algorithm largely contributes to the high computational load of the BCDPCM codec, while the pitch prediction algorithm do the same in case of the Opus codec. So alternate approaches can be devised to reduce the amount of computation.

List of Abbreviations

A	
ADC	Analog to digital conversion
B	
BCDPCM	Block companded differential pulse code modulation
C	
CD	Compact disc
Codec	Encoder and Decoder
D	
DAC	Digital to analog conversion
F	
FEC	Forward error correction
FFT	Fast fourier transform
G	
GAL	Gradient adaptive lattice
GUI	Graphical user interface
I	
IMDCT	Inverse modified discrete cosine transform
IP	Internet protocol
M	
MDCT	Modified discrete cosine transform
MOV	Model output variables
N	
NMR	Noise to mask ratio
O	
ODG	Objective difference grade
P	
PARCOR	Partial Correlation
PCM	Pulse code modulation
PEAQ	Perceptual evaluation of audio quality
PLC	Packet loss concealment

PVQ	Pyramid vector quantization

S

SMR	Signal to mask ratio
SNR	Signal to noise ratio

T

TCP	Transmission control protocol
TDAC	Time domain alias cancellation

List of Software

Name	Version	URL	Comment
libogg	1.3.0	https://www.xiph.org	Provides functionalities required by Opus
libopus	1.1	http://www.opus-codec.org	Opus source Code and libraries
opus-tools	0.1.2	http://www.opus-codec.org	Provides command line utility for Opus
valgrind	3.7.0	http://valgrind.org	Tool suite for memory checks and profiling
callgrind	-	http://valgrind.org	Profiling tool, part of valgrind tool suite
KCachegrind	0.7.4	http://kcachegrind.sourceforge.net	Profile data Visualization tool
Peaq	0.4	-	Tool for audio quality evaluation

Table 6.2: List of software.

Bibliography

[1] J. Valin, T. B. Terriberry, C. Montgomery, and G. Maxwell, “A high-quality speech and audio codec with less than 10 ms delay,” *IEEE Transactions on Audio, Speech and Language Processing*, 2010.

[2] G. Simkus, M. Holters, and U. Zölzer, “Ultra-low delay lossy audio coding using dpcm and block companded quantization,” *2013 Australian Communications Theory Workshop (AusCTW)*, January 2013.

[3] J.M.Valin, G.Maxwell, T.B.Terriberry, and K. Vos, “High-quality, low-delay music coding in the opus codec,” *135th AES Convention*, no. 5, October 2013.

[4] T. R. Fischer, “A pyramid vector quantizer,” *IEEE Transactions on Information Theory*, vol. IT-32, July 1986.

[5] J. Spittka and J. M. Valin, “Definition of the opus audio codec-draft-ietf-codec-opus-16,” *Network Working Group*, 2012.

[6] G. Simkus, M. Holters, and U. Zölzer, “Error robust delay-free lossy audio coding based on adpcm,” *In proceeding of: Digital Audio Effects - DAFX 2013*, September 2013.

[7] T. Thiede, W. C. Treurniet, R. Bitto, and C. Schmidmer, “Peaq-the itu standard for objective measurement of perceived audio quality,” *J.Audio Engineering Society*, vol. 48, 2000.

[8] R. R. Coifman and M. V. Wickerhauser, “Entropy-based algorithms for best basis selection,” *IEEE Transactions on Information Theory*, vol. 38, March 1992.

[9] M. Fink, M. Holters, and U. Zölzer, “Comparison of various predictors for audio extrapolation,” *In proceeding of: Int. Conference on Digital Audio Effects*, vol. 16, September 2013.

[10] M. Lutzky, G. Schuller, M. Gayer, U. Krämer, and S. Wabnik, “A guideline to audio codec delay,” *AES 116th Convention,paper 6062*, May 2004.

[11] A. Spanias, T. Painter, and V. Atti, “Audio Signal Processing and Coding,” 2006, 978-0-471-79147-8:ISBN.

[12] U. Zölzer, “Digital audio signal processing,2nd edition,” 2008, 978-0-470-99785-7:ISBN.

[13] B. F. Boroujeny, “Adaptive filters-theory and applications,” 1999, 0-471-98337-3:ISBN.

[14] M.Bosi and R. E. Goldberg, “Introduction to digital audio coding and standards,” 2003, 1-4020-7357-7:ISBN.