Opus General Information

Opus demo

http://xiph.org/~giles/2012/opus/

Since May 5, 2012, Firefox 15 has had support for playing Opus files in HTML <audio> and <video> elements.

Opus is a new audio codec we hope will become a new standard for audio on the web. Features include:

- Better compression than mp3/ogg/aac.
- Good for both music and spoken word.

- Dynamically adjustable bitrate, audio bandwidth, and coding delay.
- Good for real-time and pre-recorded applications.

After launching Firefox, try loading an opus file. Like this one from our test suite.

Firefox 15 became the stable release on August 28, 2012.

It's Opus, it rocks and now it's an audio codec standard!

https://hacks.mozilla.org/2012/09/its-opus-it-rocks-and-now-its-an-audio-codec-standard/

on September 11, 2012 by Jean-Marc Valin and Timothy B. Terriberry

In a great victory for open standards, the Internet Engineering Task Force (IETF) has just standardized Opus as RFC 6716.

Opus is the first state of the art, free audio codec to be standardized. We think this will help us achieve wider adoption than prior royalty-free codecs like Speex and Vorbis. This spells the beginning of the end for proprietary formats, and we are now working on doing the same thing for video.

There was both skepticism and outright opposition to this work when it was first proposed in the IETF over 3 years ago. However, the results have shown that we can create a better codec through collaboration, rather than competition between patented technologies. Open standards benefit both open source organizations and proprietary companies, and we have been successful working together to create one. Opus is the result of a collaboration between many organizations, including the IETF, Mozilla, Microsoft (through Skype), Xiph.Org, Octasic, Broadcom, and Google.

A highly flexible codec

Unlike previous audio codecs, which have typically focused on a narrow set of applications (either voice or music, in a narrow range of bitrates, for either real-time or storage applications), Opus is highly flexible. It can adaptively switch among:

- Bitrates from 6 kb/s to 512 kb/s
- Voice and music
- Mono and stereo
- Narrowband (8 kHz) to Fullband (48 kHz)
- Frame sizes from 2.5 ms to 60 ms

Most importantly, it can adapt seamlessly within these operating points. Doing all of this with proprietary codecs would require at least six different codecs. Opus replaces all of them, with better quality.

The specification is available in RFC 6716, which includes the reference implementation. Up-to-date software releases are also available.

Some audio standards define a normative encoder, which cannot be improved after it is standardized. Others allow for flexibility in the encoder, but release an intentionally hobbled reference implementation to force you to license their proprietary encoders. For Opus, we chose to allow flexibility for future encoders, but we also made the best one we knew how and released that as the reference implementation, so everyone could use it. We will continue to improve it, and keep releasing those improvements as open source.

Use cases

Opus is primarily designed for use in interactive applications on the Internet, including voice over IP (VoIP), teleconferencing, in-game chatting, and even live, distributed music performances. The IETF recently decided with "strong consensus" to adopt Opus as a mandatory-to-implement (MTI) codec for WebRTC, an upcoming standard for real-time communication on the web. Despite the focus on low latency, Opus also excels at streaming and storage applications, beating existing high-delay codecs like Vorbis and HE-AAC. It's great for internet radio, adaptive streaming, game sound effects, and much more.

Although Opus is just out, it is already supported in many applications, such as Firefox, GStreamer, FFMpeg, foobar2000, K-Lite Codec Pack, and lavfilters, with upcoming support in VLC, rockbox and Mumble.

For more information, visit the Opus website.

Opus

http://www.reddit.com/r/linux/comments/1grbe8/

Visited & extracted on 2015-05-30.

I wouldn't go quite that far. Opus does seem to be better (and definitely handles lower bitrates better, especially for human speech), but Vorbis is still pretty dang good. It's also got a pretty wide reach already, too.

I wouldn't mock anyone for continuing to encode to Vorbis for a while yet. That said, I'm switching to Opus myself already...

epicanis 4 points 1 year ago

Opus made it into the WebRTC standard and can be played in both mobile and desktop versions of Firefox / vlc and is working on my Rockbox Sansa Clip. Flac is found on high end set top boxes. Vorbis has been used and packaged behind the scenes in many games, and is part of WebM which is what Youtube encodes to.

redsteakraw 9 points 1 year ago

I moved my entire music collection to opus overnight and can play it anywhere I go.

habys 7 points 1 year ago

It's been said of vorbis, it's been said of vp3, theora, vp8.

Look at how that went. Where are those patents? Where are those lawsuits?

Yeah, right.

MPEG will soon be out of the picture. It's clear the MPEG model is obsolete. It's been demonstrated the community (mainly gathering around Xiph.org) can create royalty free, open codecs with free software implementations.

Lossless audio? FLAC.

Low bitrate, low latency audio, for streaming video and VoIP? OPUS. (absolute best at 8-128kbit/s).

General purpose lossy audio? Vorbis. (better quality/bitrate than AAC according to hydrogenaudio's massive double-blind tests. Uses less CPU. And the codec is a way simpler and more beautiful design, on top of it. Smart game developers use it.).

High quality video? Daala. (if it really manages to be better and use less cpu than HEVC and VP9). While that's not ready just yet, the royalty free VP9 is.

Low CPU requirements in order to decode video? Theora. (higher quality than divx/ffmpeg, simpler design, less cpu, smart game developers use it)

3G6A5W338E 3 points 1 year ago*

Opus, the revolutionary open audio codec for podcasts and internet audio

https://auphonic.com/blog/2012/09/26/opus-revolutionary-open-audio-codec-podcasts-and-internet-audio/

Written by flo on Sept. 26, 2012 in Audio .

We are very happy that **Opus**, an exciting new open audio codec, is now officially **standardized** by the Internet Engineering Task Force (IETF). Opus is particularly interesting for podcasts, radio books and internet audio in general due to its **suitability for both, music and speech**, and its **outstanding sound quality at also very low bitrates**.

In this article, we will listen to some examples of the Opus codec in action and have a closer look at the codec and how audio producers can benefit from it.

Why Opus?

Opus is a lossy audio codec that has some significant advantages over other lossy codecs such as MP3 or AAC. First, Opus is an open standard, and as such is **royalty-free**. Another advantage of Opus is its **remarkable audio quality**, especially at low bitrates. Moreover, this quality is achieved at **very low latencies**, which makes Opus a logical choice for interactive music and speech transmission. However, it is also very well suited for storage and streaming applications.

In fact, its developers call Opus **the swiss army knife of audio codecs** and propose it as a suitable replacement for almost all other audio codecs, with the exceptions of the losslessFLAC codec and the ultra-low-bitrate Codec2, which was designed for ham radio.

Sound Examples

Let's listen to some examples of the Opus codec in action! Jean-Marc Valin, one of the Opus developers, notes that for music, the quality of Opus at 64 kb/s compares to that of MP3 at 96 kb/s. For speech, the difference is even more pronounced.

Consider the following sound example:

Uncompressed speech (download):

At Auphonic, we offer 48 kb/s as the default value for Opus encoding, as opposed to 64 kb/s for AAC and 96 kb/s for MP3. For speech, this works rather well (note that all examples have been decoded back to *.wav* to ensure browser compatibility):

Opus at 48 kb/s (download):

For lower bitrates, the advantages of Opus become increasingly apparent. Consider the following examples, which compare Opus to MP3 at 24 kb/s:

MP3 at 24 kb/s (download):

Opus at 24 kb/s (download):

At 16 kb/s, MP3 really starts to struggle, whereas Opus still features excellent speech intelligibility (that's about 7MB per hour!):

MP3 at 16 kb/s (download): Opus at 16 kb/s (download):

At even lower bitrates, the differences become even more pronounced – here is an example of Opus at the extremely low bitrate of 6 kb/s. Of course this sounds somewhat tinny (as you would expect at 6 kb/s), but compare it with an 8 kb/s MP3 file:

MP3 at 8 kb/s (download): Opus at 6 kb/s (download):

Because of its convincing low-bitrate performance, Auphonic will offer Opus encoding all the way down to 6 kb/s, which is the lowest bitrate that Opus supports. Our hope is that this will allow media producers to **reach additional audiences in low-bandwidth areas**, giving these people access to content that they would otherwise be excluded from.

History

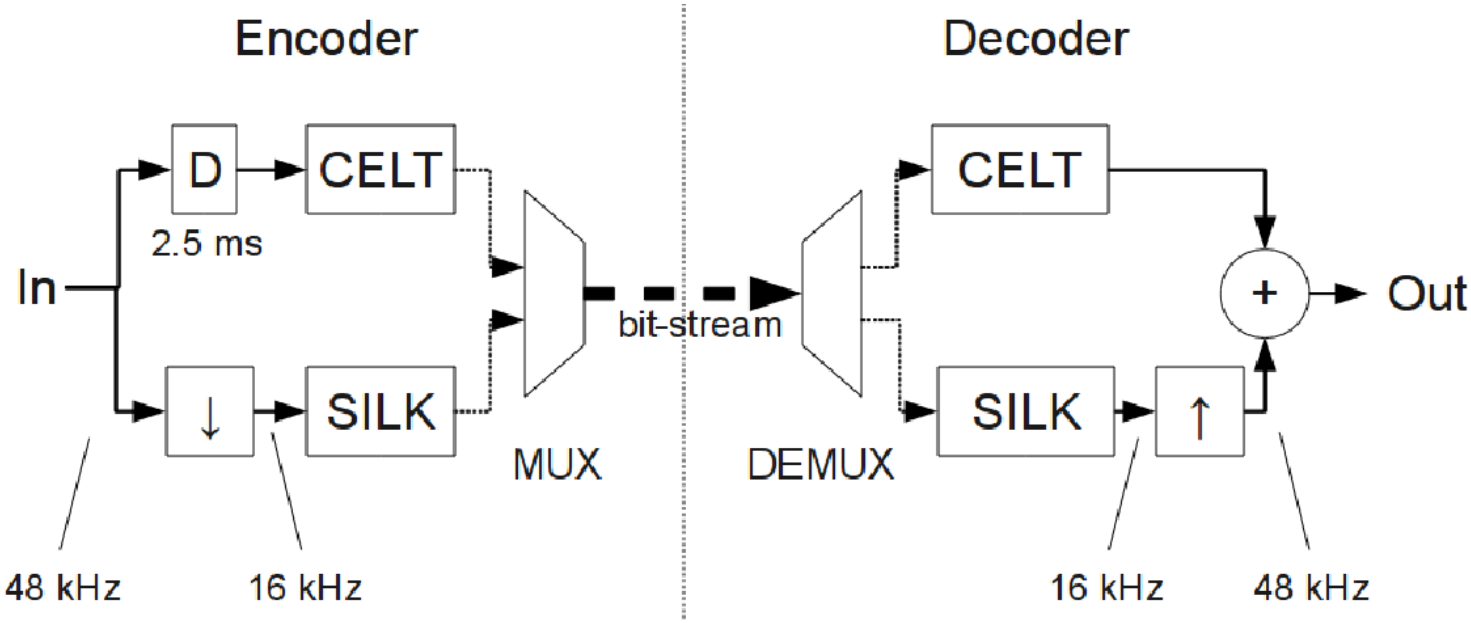
The Opus codec is based on **two originally independent development efforts**:

Xiph.org started work on a codec called **CELT** in 2007, with the intention of bridging the gap between Vorbis (their high-bitrate audio codec) and Speex (their speech codec) for applications where both high quality audio and low delay are desired. At the same time, Skype was working on **SILK**, the next-generation speech codec for their VoIP software.

After forces had eventually been joined in 2010, the Internet Engineering Task Force (IEFT)approved Opus for standardization as RFC 6716 two years later, and on 11 September 2012, the first stable versions were officially released.

How does Opus work?

What makes Opus particularly attractive for podcasting is the fact that it **unites the benefits of music- and speech-specific audio codecs**. This is made possible by running two encoders in parallel, one based on a modified CELT encoder, and the other one on an extended SILK.



The Opus encoding and decoding process.

Figure from Opus, the Swiss Army Knife of Audio codecs (p.7).

The **SILK** codec, which is based on linear predictive coding (LPC), is primarily designed for speech transmission at low latencies, at the expense of not being very suitable for music. The**CELT** codec, on the other hand, is a fullband general-purpose codec based on the modified discrete cosine transform (MDCT) and targets both, speech and music, at higher bitrates.

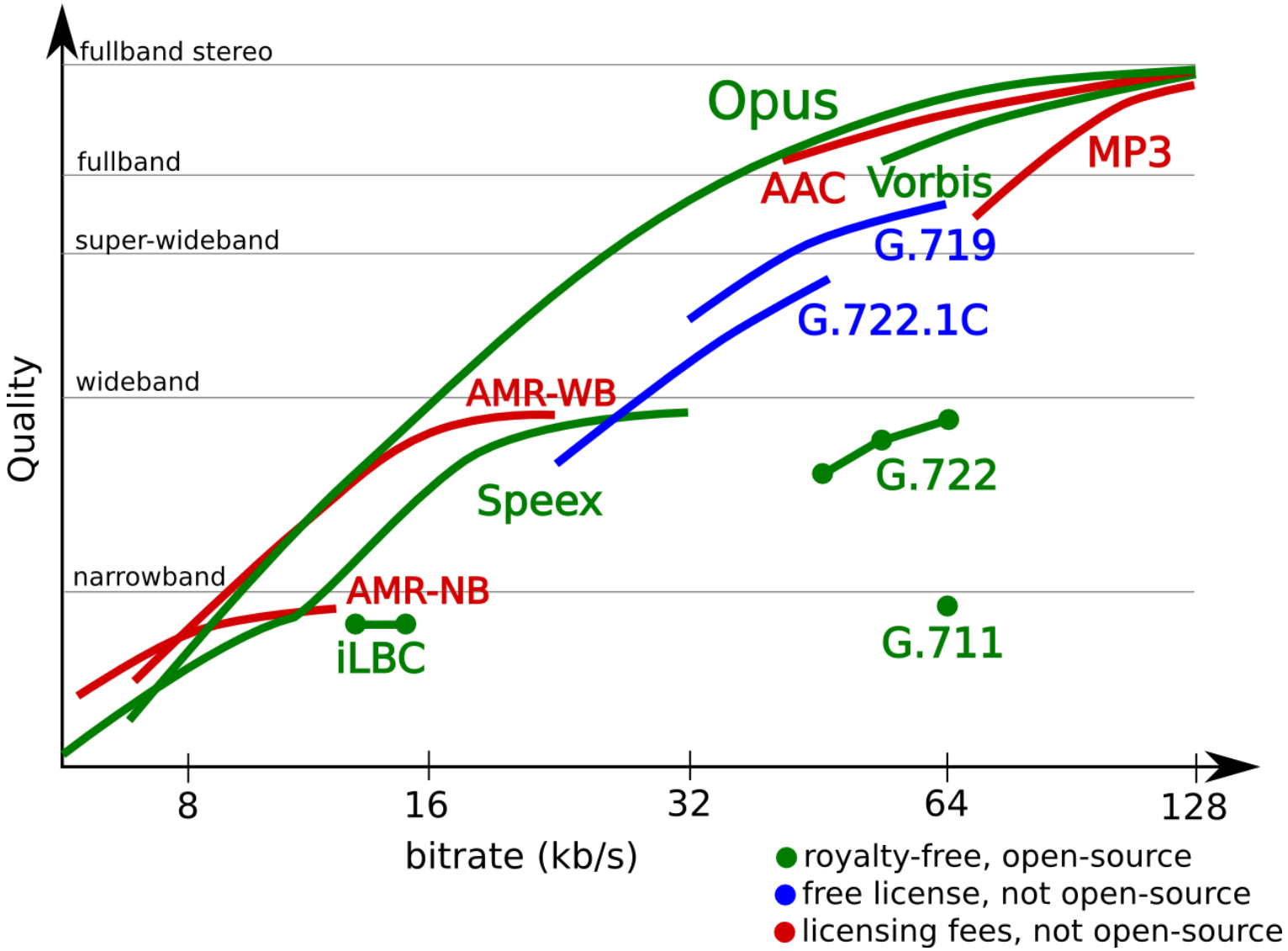
To get the best of both worlds, Opus combines these two codecs in three different modes, which Jean-Marc Valin describes as follows:

- The **SILK only** mode is particularly suitable for speech from lowband to wideband.
- The **CELT only** mode is the logical choice for encoding music.
- Opus also provides a **hybrid mode** that uses CELT for high frequencies (> 8kHz) and SILK for low frequencies (< 8kHz). This mode can be used to encode speech at a higher quality than the SILK-only mode can provide.

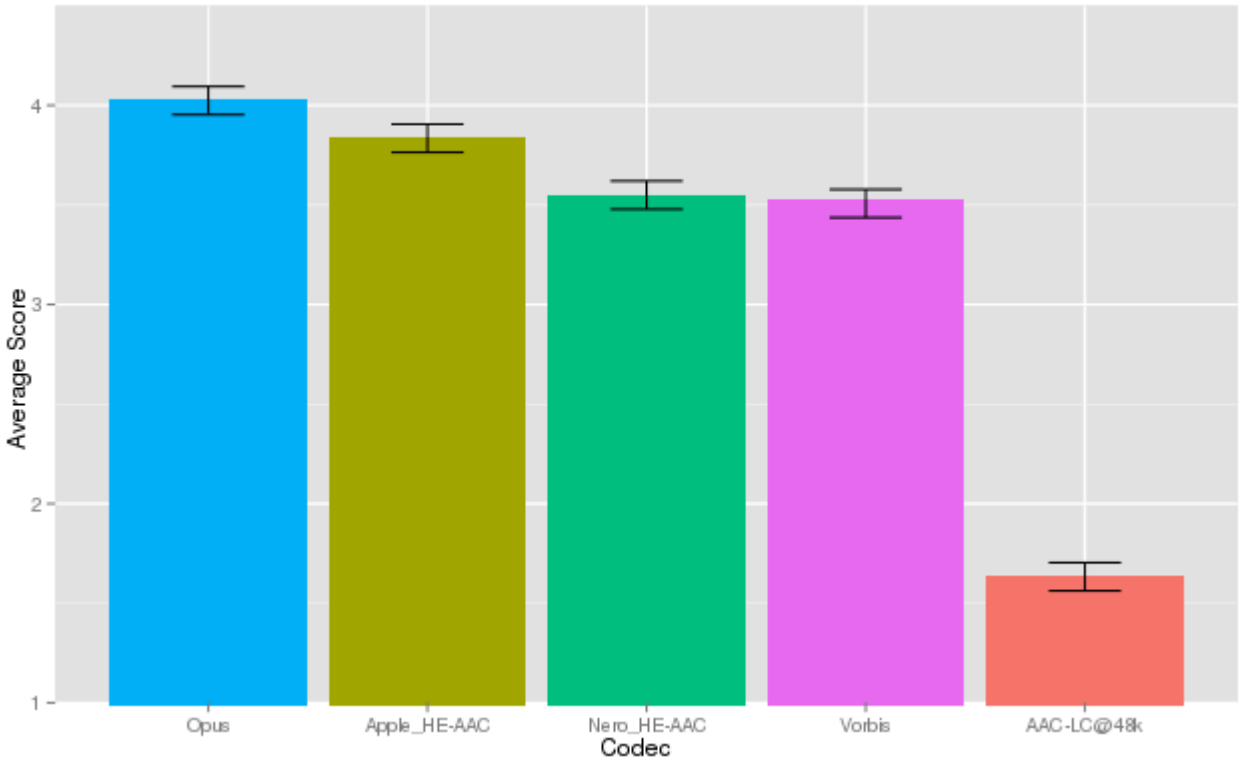
Which mode is actually used is determined implicitly through the chosen bitrate (which, by the way, can be changed on-the-fly). Future versions of the encoder will allow music/speech detection algorithms to automatically inform the choice of mode. At Auphonic, we already concern ourselves a lot with automatic music and speech detection, so our users should be able to benefit directly from such developments.

How does Opus compare to other Codecs?

The Opus codec has been subjected to a number of independent listening tests, conducted byHydrogen Audio, Google (documentation available here and here), and Nokia, all with very convincing results regarding the performance of Opus compared to other codecs.



Opus quality compared to other codecs. Figure from the Opus website.



Results of the Hydrogen Audio listening test comparing Opus to other codecs.

Figure from the Xiph website.

How can I try Opus?

Having been released only very recently, Opus is not yet widely supported. Considering its qualities, one can expect support to eventually spread quite rapidly, but let's have a look at what's already available today:

- If you are merely interested in **listening** to further examples of what the codec sounds like, the Opus website provides some excellent and easy-to-compare samples and interactive demos of Opus-encoded speech and music at different bitrates.
- If you have some Opus-encoded material that you need to **play**, the popular audio player foobar2000 for Windows includes an Opus decoder in its latest stable release. The cross-platform VLC player will decode Opus files in its upcoming 2.0.4 release, Android users can already listen to Opus using VLC Beta for Android (Version 0.0.4 or higher). Firefox (and Thunderbird) supports Opus from version 15, as all WebRTC-compatible browsers will eventually have to.
- If you actually want to **encode** some of your own audio material, we suggest you try Opus with your next Auphonic production. Experimentally minded folks can also try the command-line program *opusenc* in the *opus-tools* package. Opus-encoded audio can be encapsulated in Ogg containers, using the *.opus* file extension. The encapsulation of Opus data in Matroska containers is currently under development.

Summary

Opus is an outstanding codec for lossy audio compression regarding a wide range of applications:

- Because it is **suitable for music and speech**, it is particularly interesting for podcasts, audio books, radio shows and internet audio.
- It **performs really well also at very low bitrates**, allowing producers to reach audiences also in areas with low connectivity.
- Because Opus is an **open standard**, producers will benefit directly from further developments of the codec. Widespread support for Opus among music players and other audio applications can be expected for the near future.

At Auphonic, we are excited to adopt Opus already at this early stage and hope to contribute to its acceptance as the versatile and wonderful codec that it is.

Opus bandwidth used internally

http://www.hydrogenaud.io/forums/index.php?showtopic=105326&view=findpost&p=862871

Here are the relevant internal definitions lifted from the spec. Whichever bandwidth type is used internally, the sample rate is always reported externally as 48 kHz, and timings, samples, etc are all calculated using this rate.

CODE

```
+----------------------+-----------------+-------------------------+
  | Abbreviation         | Audio Bandwidth | Sample Rate (Effective) |
  +----------------------+-----------------+-------------------------+
  | NB (narrowband)      |          4 kHz  |                  8 kHz  |
  |                      |                 |                         |
  | MB (medium-band)     |          6 kHz  |                 12 kHz  |
  |                      |                 |                         |
  | WB (wideband)        |          8 kHz  |                 16 kHz  |
  |                      |                 |                         |
  | SWB (super-wideband) |         12 kHz  |                 24 kHz  |
  |                      |                 |                         |
  | FB (fullband)        |      20 kHz (*) |                 48 kHz  |
  +----------------------+-----------------+-------------------------+
```

Each of the five bandwidths is only available with certain encoding modes (Celt/Silk/hybrid) and certain frame sizes, giving a total of 32 possible audio frame configurations.

Opus Encoding

Take away --

- *All Opus Encoding settings already default to the recommended value, so the only option to set might be the bitrate.*
- *Opus has very high speech quality even with 32 kb/s bitrate.*
  The default complexity is 10 for ffmpeg
- *The vbr is default for opusenc*

Opus Encoder

https://mf4.xiph.org/jenkins/view/opus/job/opus/ws/doc/html/group__opus__encoder.html#details

It is possible to change some of the encoder's settings using the **opus_encoder_ctl()** interface. *All these settings already default to the recommended value, so they should only be changed when necessary*. The most common settings one may want to change are:

```
opus_encoder_ctl(enc, OPUS_SET_BITRATE(bitrate));
opus_encoder_ctl(enc, OPUS_SET_COMPLEXITY(complexity));
opus_encoder_ctl(enc, OPUS_SET_SIGNAL(signal_type));
```

where

- bitrate is in bits per second (b/s)
- complexity is a value from 1 to 10, where 1 is the lowest complexity and 10 is the highest
- signal_type is either OPUS_AUTO (default), OPUS_SIGNAL_VOICE, or OPUS_SIGNAL_MUSIC

Guidelines for high quality audio encoding

https://trac.ffmpeg.org/wiki/Encode/HighQualityAudio

- Avoid transcoding from a lossy format to the same or another lossy format when possible. Transcode to lossy from the lossless source (if you have it), or just copy the lossy source audio track instead of transcoding.
- Transcoding from a lossy format like MP3, AAC, Vorbis, Opus, WMA, etc. to the same or different lossy format might degrade the audio quality even if the bitrate stays the same (or higher).
- FFmpeg can encode to a wide variety of lossy audio formats. Based on quality produced from high to low:
  `libopus > libfdk_aac = libvorbis > libmp3lame >= libfaac >= eac3/ac3 > aac > libtwolame > vorbis > mp2 > wmav2/wmav1 > libvo_aacenc`

Recommended minimum bitrates to use

The bitrates listed here assume 2-channel stereo and a sample rate of 44.1kHz or 48kHz. Mono may require fewer bits.

- **libopus** Usable range >= 80Kbps. Recommended range **>= 128Kbps**

Does Opus support higher sampling rates?

https://wiki.xiph.org/OpusFAQ#Does_Opus_support_higher_sampling_rates.2C_such_as_96_kHz_or_192_kHz.3F

Yes and no.

Opus encoding tools like opusenc will happily encode files that are sampled at 96 or 192 kHz.

However, input files at these rates are internally **converted to 48 kHz** and then only frequencies **up to 20 kHz** are encoded.

The reason is simple: lossy codecs are designed to preserve audible details while discarding irrelevant information. Since the human ear can only hear up to 20 kHz at best (usually lower than that), frequency content above 20 kHz is the first thing to go.

See Monty's 24/192 Music Downloads ...and why they make no sense for more details.

Why not keep the SILK and CELT codecs separate?

Opus is more than just two independent codecs with a switch.

In addition to a linear prediction "SILK mode" and a MDCT "CELT mode" it has a "hybrid mode," where speech frequencies up to 8 kHz are encoded with LP while those above 8 kHz are encoded with MDCT. This is what allows Opus to have such high speech quality around 32 kb/s.

Another advantage of the integration is the ability to switch between these modes seamlessly, without any "glitch" and without any out-of-band signalling.

How to encode audio with Opus codec?

http://superuser.com/questions/516806/how-to-encode-audio-with-opus-codec

> *The Opus audio codec looks like the best thing ever for compressing audio. It has recently become supported in the latest ffmpeg and VLC players. However, there is no documentation I can find on how to actually encode media with it. Can someone please direct me to said docs, preferably with specifics to ffmpeg flags and usage? I have a lot of audiobooks that are taking up far too much space and Opus looks like the perfect format to keep them in.*

```
ffmpeg -i input -acodec libopus -b:a <bitrate> -vbr on -compression_level 10 output
```

The ffmpeg documentation has a list of options and descriptions for libopus.

Make sure you compiled ffmpeg with `--enable-libopus`!

`compression_level (comp)`

> Set encoding algorithm complexity. Valid options are integers in the 0-10 range. 0 gives the fastest encodes but lower quality, while 10 gives the highest quality but slowest encoding. The default is 10.

Dec 5 '13

Adam Chance

1. Download Opus-tools
2. Encode:
3. `opusenc --bitrate 64 What_A_Feeling.wav What_A_Feeling_64.opus`
4. Decode: (to play in any media player, useful if your media player does not support opus yet):
5. `opusdec What_A_Feeling_64.opus What_A_Feeling_opus64.wav`

(What_A_Feeling is a song name)

Detailed options displayed when running `opusenc` by itself:

Dec 9 '12

Alexey Eromenko

How to convert a sound file to Opus

http://askubuntu.com/questions/211054/how-to-convert-a-sound-file-to-opus

> *I want to convert MP3 and WAV files to Opus, what are the steps?*

Per default the audio converter shipped with the opus-tools can convert audio in raw, wave or AIFF format. The minimal syntax uses default settings:

```
opusenc input.wav output.opus
```

We may want to add a better bitrate as the default 96 kbps with the option `--bitrate N.nnn` (for all options consult the manpage for opusenc).

To convert mp3 "on the fly". i.e. without creating a temporary file we can pipe the output from avconv to opusenc like this:

```
avconv -i input.mp3 -f wav - | opusenc --bitrate 256 - output.opus
```

answered Nov 2 '12

Takkat

I have found by far the best solution is the one that has become available very recently: compile the opus audio encoder and decoder as noted here, and build `ffmpeg` with opus support by adding `--enable-opus` to the configure options of `ffmpeg` (as listed on the compilation guide).

answered Jan 10 '13

user76204

Ubuntu 14.04 and Debian 8 ship with version 9 of `libav-tools` in their repositories, and it has built-in support for Opus through the package `libopus0`.

Example 1: Reencode an audio file as opus

With version 9 of `libav-tools` and `libopus0` installed you can simply, for example, do:

```
avconv -i file.mp3 -map 0:a -codec:a opus -b:a 100k -vbr on file.opus
```

What the options do

- `-i file.mp3` sets the input file.
- `-map 0:a` will select all audio streams (`a`) from the input file `0`. Read more about `-map` on https://libav.org/avconv.html#Advanced-options
- `-codec:a opus` selects the opus encoder for the audio streams (`a`). Read more about `-codec` on https://libav.org/avconv.html#Main-options.
- `-b:a 100k` sets the audio's bitrate to 100 kilobit/s. Read more about `-b` onhttps://libav.org/avconv.html#Codec-AVOptions
- `-vbr on` turns on variable bitrate. This is an option specific for libopus. The `avconv -h full` gives all options for libopus.
- `file.opus` sets the output file.

Example 2: Grab the audio from a video file and encode it as opus

Take the second stream of the first input (`-map 0:1`), which is the audio stream. Encode it with libopus at 100 kbit/s with variable bitrate on:

```
$ avconv -stats -i linuxactionshowep309-432p.mp4 -map 0:1 -c libopus -b 100k
linuxactionshowep309-432p-audio-only.opus
```

With the package `mediainfo` installed:

```
$ mediainfo linuxactionshowep309-432p-audio-only.opus
```

VBR Encoding with libopus

http://ffmpeg-users.933282.n4.nabble.com/VBR-Encoding-with-libopus-td4657230.html

> *libopus lets us supply -vbr with the values "off", "on" and "constrained".*
> *In the latter two modes, how does one choose the quality level?*
>
> *-q:a does not seem to have any effect, and Opus falls back to 96 kBit/s.*
> *Without any option other than -vbr on, it also uses CBR encoding.*
>
> *Furthermore, the source code (*
> *https://github.com/FFmpeg/FFmpeg/blob/master/libavcodec/libopusenc.c)*
> *mentions a "compression_level" option, but how can I set this as a user?*
>
> *ffmpeg -i in.mp4 -c:a libopus -vn -vbr on out.opus*
> *ffmpeg version 1.1.2 Copyright (c) 2000-2013 the FFmpeg developers*
> *built on Feb 8 2013 22:55:29 with Apple LLVM version 4.2*

The impression I have - mostly from the opusenc man page - is that opusenc (and therefore libopus) doesn't have a VBR quality-targeting mode in the same way that libmp3lame or libvorbis do.

I mean that it doesn't work the same way: with libopus, you simply provide a target bit rate, and the encoder tries to use that as the average bit rate. It's confusing because you normally only supply a bit rate for the CBR mode of a given encoder, but libopus just works differently, I think.

opusenc man page:
https://mf4.xiph.org/jenkins/view/opus/job/opus-tools/ws/man/opusenc.html

Liam Condron-Farnos, Feb 10, 2013

Opus

http://www.hydrogenaud.io/forums/index.php?showtopic=106497

I'm curious about Opus VBR mode, every thing I throw at it, the bitrate tends around my target bitrate very tightly +-5%, unlike say vorbis or lame which wildly fluctuate, 'ball park' style bitrates.

Here is an extreme example:

CODE

```
sox -n -t sox - synth 30 brownnoise brownnoise gain -3 lowpass 180 | ffmpeg -f sox -i - -q:a 10
-y output.ogg
```

This generates 30 seconds of brownnoise with a lowpass filter over the top, sounds like the warp engine of the Enterprise.

Basically Ogg vorbis at quality 10, nominal 500kbps~, but the actual output is only 183.8kbps because of the simplicity of the audio sample.

Opus on the other hand:
CODE

```
sox -n -t sox - synth 30 brownnoise brownnoise gain -3 lowpass 180 | ffmpeg -f sox -i - -b:a 510k
-vbr on -y output.opus
```

The output is 499.2kbps, this is not exactly what I expected vbr mode to do.

To me this looks like ABR, does Opus' VBR just need more time to mature?

opusenc does the same thing, at least on my specific example, I tried running a movie soundtrack through opusenc and got a bitrate of about 60kbps for 64kbps, but the 'instant bitrates' were 30-130kbps, Still find it odd that it allocates so much to such a similar sample.

#3

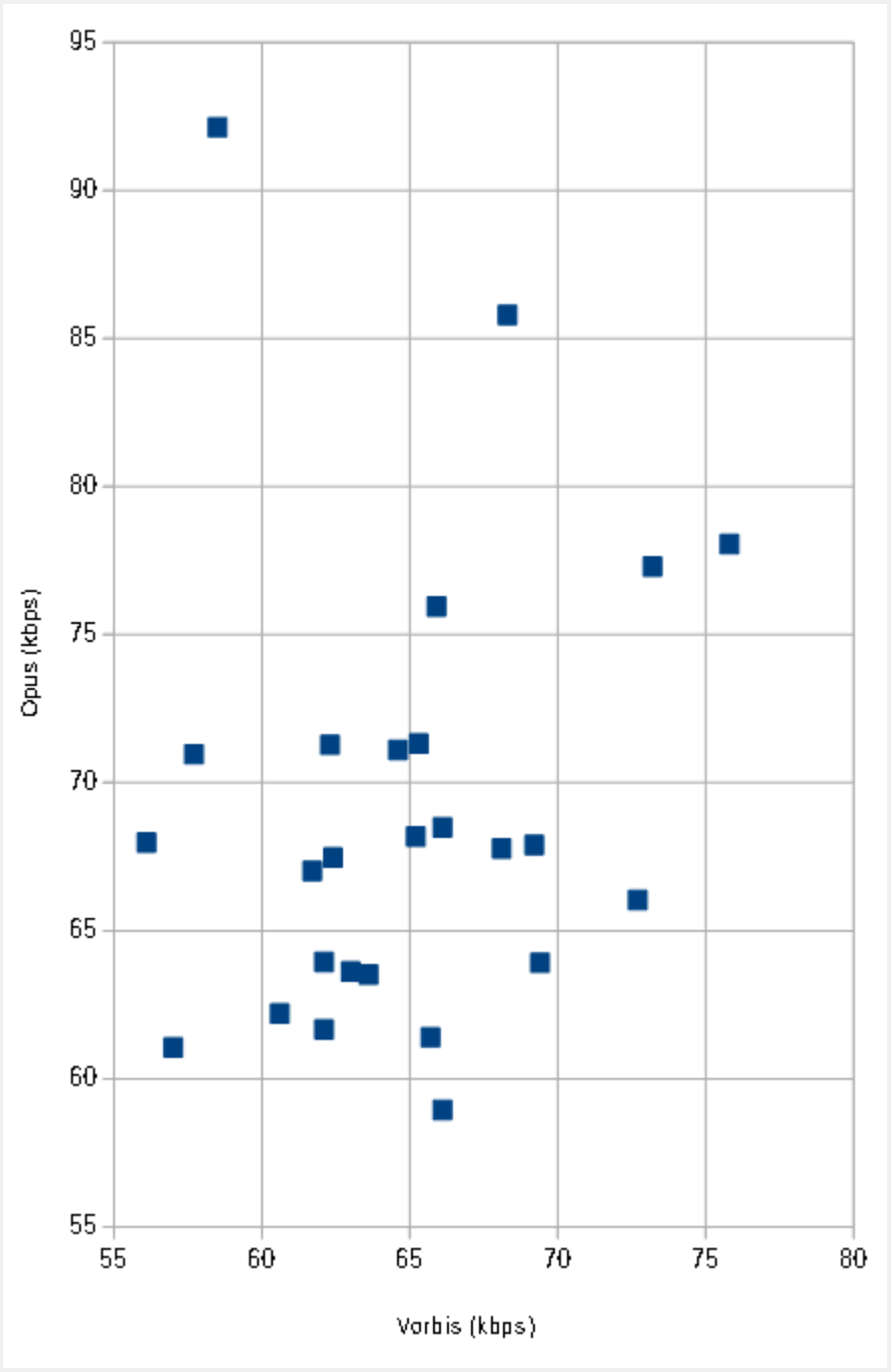
```
-b:a 510k
```

Opus has a hard limit of about 510kbps for stereo audio, which impedes the VBR algorithm. You need to use a much lower bitrate (with the latest libopus) to see the full effect.

#6

Opus is not at all like Vorbis, so it's no surprise that the VBR behavior differs between the two. Try encoding something else, you might be surprised by the results.

Here's a graph of the bitrates of each track from two CDs encoded at a target bitrate of 64kbps. I chose these CDs specifically for their variety of genres, so these results might be exaggerated somewhat. However, it should be apparent that there is (almost?) no correlation between the VBR behaviors of these codecs.



Bonus: if you can guess what type of music the top-left point represents? -- Classic or some 8 bit computer game music

VP9 Encoding Guide

http://wiki.webmproject.org/ffmpeg/vp9-encoding-guide

All three recommended settings have the same audio encoding method:

```
-c:a libopus -b:a 64k
```

c:a libopus tells FFmpeg to encode the audio in Opus.
b:a 64k tells FFmpeg to encode the audio with a target of 64 kilobits.