The Opus Codec

Technical Plenary
IETF 87
Berlin, DE
July 29, 2013

Jean-Marc Valin, Greg Maxwell, Peter Saint-Andre, Timothy B. Terriberry, Emil Ivov, Lorenzo Miniero, Justin Uberti
Outline

- Remote Participation Experiment
- Overview of Opus
- Testing
- CODEC WG History and Lessons Learned
- Future work
- Opus deployment panel
IETF Remote Participation

- Meetecho provides remote participation to IETF sessions
  - http://ietf87.conf.meetecho.com/
- Tutorial:
- Conference room associated with a session
  - Audio from the physical room mixer
  - Video from a webcam
- Active participants (can contribute to the mix)
  - Java Applet, WebRTC, Softphones, PSTN
- Passive participants (can only watch/listen)
  - Conference mix made available as a stream
    - RTSP, RTMP, HTML5
Opus Experiment (Live Now!)

- Remote participation for this technical plenary:
  - http://www.meetecho.com/ietf87/tech_plenary
- For information on remote participation and additional links relating to OPUS, please check the IAB wiki: http://trac.tools.ietf.org/group/iab/trac/wiki/IETF-87
- WebRTC-only setup available for remote speakers
  - Asterisk+Opus mixing audio at 48kHz
  - Open source MCU switching video feeds
    - http://lynckia.com/
- Have something to say?
  - Raise your hand! (well, maybe later)
Outline

- Remote Participation Experiment
- **Overview of Opus (Jean-Marc Valin)**
- Testing
- CODEC WG History and Lessons Learned
- Future work
- Opus deployment panel
What is Opus?

- Audio codec designed for interactive Internet application
- Published as RFC 6716 in Sept 2012
- Works for most audio applications
- Adopted as MTI codec for WebRTC
Why a New Audio Codec?

http://xkcd.com/927/
http://imgs.xkcd.com/comics/standards.png
Why a New Audio Codec?

- No pre-existing audio codec that would:
  - Provide good audio quality over the Internet
  - Be published as a standard
  - Be freely implementable
# Two types of audio codecs

<table>
<thead>
<tr>
<th>Speech codecs</th>
<th>Audio codecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice communication</td>
<td>Music streaming/storage</td>
</tr>
<tr>
<td>Low delay</td>
<td>High delay</td>
</tr>
<tr>
<td>Narrowband-Wideband</td>
<td>Fullband</td>
</tr>
<tr>
<td>“Toll quality”</td>
<td>High Quality</td>
</tr>
<tr>
<td>G.729, AMR, Speex</td>
<td>MP3, AAC, Vorbis</td>
</tr>
</tbody>
</table>

- We want (and can now afford) the best of both worlds
Applications and Standards (2010)

<table>
<thead>
<tr>
<th>Application</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoIP with PSTN</td>
<td>AMR-NB</td>
</tr>
<tr>
<td>Wideband VoIP/videoconference</td>
<td>AMR-WB</td>
</tr>
<tr>
<td>High-quality videoconference</td>
<td>G.719</td>
</tr>
<tr>
<td>Low-bitrate music streaming</td>
<td>HE-AAC</td>
</tr>
<tr>
<td>High-quality music streaming</td>
<td>AAC-LC</td>
</tr>
<tr>
<td>Low-delay broadcast</td>
<td>AAC-ELD</td>
</tr>
<tr>
<td>Network music performance</td>
<td></td>
</tr>
</tbody>
</table>
## Applications and Standards (2013)

<table>
<thead>
<tr>
<th>Application</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoIP with PSTN</td>
<td>Opus</td>
</tr>
<tr>
<td>Wideband VoIP/videoconference</td>
<td>Opus</td>
</tr>
<tr>
<td>High-quality videoconference</td>
<td>Opus</td>
</tr>
<tr>
<td>Low-bitrate music streaming</td>
<td>Opus</td>
</tr>
<tr>
<td>High-quality music streaming</td>
<td>Opus</td>
</tr>
<tr>
<td>Low-delay broadcast</td>
<td>Opus</td>
</tr>
<tr>
<td>Network music performance</td>
<td>Opus</td>
</tr>
</tbody>
</table>
Specifications

- Highly flexible
  - Bit-rates from 6 kb/s to 510 kb/s
  - Narrowband (8 kHz) to fullband (48 kHz)
  - Frame sizes from 2.5 ms to 60 ms
  - Speech and music support
  - Mono and stereo
  - Optional forward error correction (FEC)

- All changeable dynamically with in-band signalling
Implementation

- Available for floating-point and fixed-point
- Wide range of supported platforms
  - x86, ARM, MIPS, SPARC, VAX, ...
- Arch-specific optimization on x86, ARM
- Quality vs complexity trade-off
- Support for packet-loss concealment (PLC) and discontinuous transmission (DTX)
Optimized for the Internet?

- More than the ability to conceal lost packets
- Wide range of operating conditions (delay, bit-rate, loss) that vary with time
- Transports data in bytes
- RTP payload: the simpler the better
How it Works

- Merge of two technologies
  - SILK: Skype's linear prediction speech codec
  - CELT: Xiph.Org's low-delay transform codec
- Better than the sum of the parts
  - Hybrid mode
  - Mode switching
Adoption

- VoIP/videoconference
  - WebRTC (Firefox, Chrome)
  - Many VoIP clients (Jitsi, Meetecho, CounterPath)
  - Games (Mumble, TeamSpeak)

- Players
  - HTML5 (Firefox, Chrome*)
  - Standalone (Rockbox, VLC, Foobar 2k)

- Network music performances
- Streaming (icecast)
Outline

- Remote Participation Experiment
- Overview of Opus
- Testing (Greg Maxwell)
- CODEC WG History and Lessons Learned
- Future work
- Opus deployment panel
Testing Opus

- Opus has a broad scope
  - 64 configurations = 4096 configuration transition pairs
  - At 1275 bitrates (in CBR alone)

- Multiple testing objectives
  - Development testing
  - Quality and bitrate targets: “Better than” Speex, iLBC, G.722.1, G.722.1C (RFC 6366)

- Used both subjective and objective testing
Subjective results

- draft-ietf-codec-results-03
  - Four different testing parties on the final codec
  - Seven more on pre-final bitstreams

- Some highlights:
  - **Google tests**
    - Speech at multiple rates
    - Main tests included 6 samples, 17 listeners
    - BS.1534-1 “MUSHRA”
  - **HydrogenAudio**
    - 64kbit/sec stereo music
    - 30 samples, 33 listeners, 531 final measurements
    - BS.1116-1 “ABC/HR”
Google results

Narrowband/Wideband/Fullband

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>LP 3.5</th>
<th>LP 7</th>
<th>iLBC 15</th>
<th>Opus 11</th>
<th>Speex 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Narrowband</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wideband/Fullband</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Values:
- Original: 99.6
- LP 3.5: 53.1 ± 2.1
- LP 7: 88.8 ± 2.2
- iLBC 15: 48.6 ± 2.3
- Opus 11: 55.6 ± 2.4
- Speex 11: 45.5 ± 2.5

Values:
- Original: 99.3
- 6.71932: 68.5 ± 2.5
- Opus 32: 97.2 ± 2.6
- Speex 24: 45.5 ± 2.7
- Opus 20: 77.9 ± 2.8
- LP 7: 72.9 ± 2.9
- G.222.1.24: 48.9 ± 2.10
- AMR-WB 20: 66.6 ± 2.11
- LP 3.5: 37.9 ± 2.12
HydrogenAudio results

The image contains a graph and a table with the following data:

Graph:
- X-axis: Vorbis, Nero HE-AAC, Apple HE-AAC, Opus
- Y-axis: Average score from 3.2 to 4.2
- Each category has a bar graph showing the average score with error bars.

Table:
| Sample | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Opus   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Apple HE-AAC |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Nero HE-AAC |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Vorbis |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Why we need more than formal listening tests

- Formal listening tests are expensive, meaning
  - Reduced coverage
  - Infrequent repetition

- Insensitivity
  - “Everything tied!”
  - Even major errors may only rarely be audible
  - Can’t detect matched encoder/decoder errors
  - Can’t detect underspecified behavior (e.g., “works on my architecture”)
Operational Testing

• Deployed to millions of users as part of Mumble, Skype, ...
  – “It sounds good except when there’s just bass”
  – “It sounds bad on this file”
  – “Too many consecutive losses sound bad”
  – “If I pass in NaNs things blow up”
Objective Quality Testing

- Run thousands of hours of audio through the codec with many settings
  - Used a 160 core cluster
  - Can run the codec 6400x real time
  - 7 days of computation is 122 years of audio
The Opus spec is executable...

- That lets us test in many different ways:
  - Operational testing
  - Objective quality testing
  - Unit testing (including exhaustive component tests)
  - Range coder mismatch testing
  - Static analysis
  - Instrumentation
  - Line and branch coverage analysis
  - White- and blackbox “fuzz” testing
  - Multiplatform testing
  - Implementation interoperability testing
Outline

- Remote Participation Experiment
- Overview of Opus
- Testing
- CODEC WG History and Lessons Learned (Peter Saint-Andre)
- Future work
- Opus deployment panel
“Storming”
(IETF 75, Stockholm)
“Forming” (IETF 76, Hiroshima)

- A much more civilized conversation :-)  
- Still skepticism about feasibility  
- But a willingness to try  
- A sense that even if we failed, we’d learn something interesting
“Norming”

- RFC 6366: Requirements for an Internet Audio Codec (August 2011)
- RFC 6569: Guidelines for Development of an Audio Code within the IETF (March 2012)
- Expectations set about IPR disclosures (cf. RFC 6702) - 13 received, all of them timely
“Performing”

- Melding the two primary contributions (CELT and SILK) went surprisingly well
- Working together on common code gave a sense of shared purpose / enterprise
- However, participants not working on the code might have felt like they were on the outside looking in
Early Sources of Confusion

- One codec or many?
- Developing something new or selecting an existing technology?
- What does it mean to be “optimized for the Internet”?
- What are the preferred IPR terms?
“Those Who Fail to Plan Are Planning to Fail”

- Have a plan for managing liaison relationships
- Have a plan for testing and for using the results to improve the codec
- Have a plan for producing an unencumbered technology
The Joys of Running Code

- Arguments over code efficiency can distract from the main purpose
- What’s the relationship between the codec and the signaling plane? (Lesson: use signaling where that would help...)
- Treating source code as normative makes typical IETF reviews more difficult
Stumbling Towards Ecstasy

• Did the WG succeed despite itself?
• In part: plenty of room for improvement if we do something similar again
• Critical to have a group of well-informed, passionate contributors with common goal
• Most important, the results are great and Opus sounds wonderful!
Outline

• Remote Participation Experiment
• Overview of Opus
• Testing
• CODEC WG History and Lessons Learned
• Future work (JM Valin & Tim Terriberry)
  • Opus
  • Video
• Opus deployment panel
Specifications

- Defining payloads
  - RTP
  - Ogg
  - Matroska
- Minor fixes to RFC 6716
Implementation

- Upcoming libopus 1.1 release
  - Fully compatible with RFC
  - Quality improvements
  - Surround improvements
  - Speech/music detection
  - Optimizations (72% faster decoder on ARM)
- libopus 1.1-beta demo:
  http://people.xiph.org/~xiphmont/demo/opus/demo3.shtml
Adoption

- **Broadcast**
  - Broadcast equipment (Tieline)
  - Digital radio (DRM, DAB)
  - Testing (EBU)

- **Internet radio**
  - http://dir.xiph.org/by_format/Opus

- **Wireless audio**
  - Speakers, microphones
Case Study: WebRTC MTI

- Mandatory To Implement (MTI) Audio Codec(s)
  - Concrete proposal (Opus+G.711) raised and decided
    - In a single meeting (IETF-84 in Vancouver)
    - Near-unanimous consensus

- Mandatory To Implement (MTI) Video Codec(s)
  - Debated heavily for over two years
  - Decision postponed at least 2 times (so far)
  - No resolution in sight
Why Was Audio So Much Easier?

- Opus produced by open, multistakeholder standardization effort
  - Including 3 of the 4 major browser vendors

- Royalty-free licensing with clear IPR history
  - Specific disclosures => easily evaluated

- And maybe... it wasn’t so easy
  - Product of 3 years of vigorous debate
  - But all that time spent making *forward* progress
Doing the same for video

- Xiph.Org Foundation’s Daala project
  - https://xiph.org/daala/
  - “Coding Party” in May
    - 169 commits from 14 authors
    - Including “individuals” from Xiph.Org, Mozilla, Cisco, Red Hat, Debian, RDIO, Voicetronix, etc.
  - Demos
    - https://people.xiph.org/~xiphmont/demo/daala/demo1.shtml
    - https://people.xiph.org/~xiphmont/demo/daala/demo2.shtml

- IETF effort
  - Bof @ IETF-85
  - List: video-codec@ietf.org
  - Drafts: draft-terriberry-codingtools, draft-egge-videocore-tdlt, draft-valin-videocore-pvq, draft-terriberry-ipr-license
Opus Deployment Panel

Timothy B. Terriberry, Mozilla/Xiph.Org Foundation: Opus in Firefox (and other places)
Justin Uberti, Google: Opus Deployment at Google
Emil Ivov, Jitsi: Audio codecs in Jitsi
Lorenzo Miniero, MeetEcho: Opus Integration in Asterisk
Opus in Firefox

- <audio> tag support in Firefox 15 (Aug. 2012)
  - Firefox 17 (Nov. 2012): Multichannel support
  - Firefox 18 (Jan. 2013): Metadata API
  - Firefox 20 (Apr. 2013): Chained streams
- WebRTC support in Firefox 22 (Jun. 2013)
  - In project branch since Aug. 2012
  - Currently mono-only (limitation of capture, AEC)
- MediaRecorder API in Firefox 25 (Oct. 2013)
  - [https://bugzilla.mozilla.org/show_bug.cgi?id=896935](https://bugzilla.mozilla.org/show_bug.cgi?id=896935)
- Music App support in Firefox OS 1.1 (release TBD)
Opus in other places

- VLC 2.0.4 (Oct. 2012, thanks to Greg Maxwell)
  - Album art support in 2.1.0 (forthcoming)

- libopusfile
  - Simple decode/playback library
  - Handles seeking, metadata, multichannel, chaining
  - Pluggable I/O backends (FILE, memory, http[s])
  - In Debian testing, Fedora 18, FreeBSD, homebrew, etc.
  - Used by: xmms2, qmmp, cmus, taglib, sox, ioquake, more...
Chrome: Initial Work

- OPUS is a very general codec with a wide range of parameters and tools.
- Integrator needs to think through which configurations it wants to support.
- Had to also solve a few integration complexities in Chrome:
  - Determination of default params
  - 48K sampling rate
  - Integration with Chrome NetEQ
Chrome Timeline

- **May 2012**
  Initial sketches on integration
- **September 2012**
  Integration started
- **October 2012**
  Working implementation
- **November 2012**
  License concerns resolved
- **December 2012 (Chrome 25)**
  Opus fully enabled in WebRTC
Chrome Timeline (cont’d)

- February 2013
  Chrome-Firefox interop demo with Opus
- March 2013 (Chrome 27)
  Opus becomes the default codec in WebRTC
- July 2013
  Opus + WebRTC used for remote participation at IETF
Chrome: Current Day

Continuing to test and improve:

- Use of Opus as default pointed out super-wideband issues in Chrome echo canceller
- Complexity on mobile CPUs needs tuning
- Proper FEC at all bitrates is not trivial
audio codecs in Jitsi

history  evolution  goals  dilemmas
then Opus happened

« totally open, royalty-free, highly versatile audio codec »
things we love in Opus

quality, usability, stereo, fullband, packet loss concealment (plc), forward error correction (fec), surround, variable bit-rate ... or not, music audio detect, manually controllable bitrate,

born at the IETF
Integrating Opus (1)

- First step was to provide lightweight integration
  - Opus encoded HTML5 stream
  - Available since IETF85 in Atlanta
- Open source setup
  - Asterisk providing mixed audio signals...
    - ... opusenc encodes the audio...
    - ... oggfwd forwards it to the streamer...
    - ... Icecast does HTML5 streaming

Passive Audio/Video (streaming) integrated in the browser by means of HTML5

- HTML5-based Audio-only stream (Opus):
  - Please beware that this stream has about 10 seconds of delay.
  - HTML5: Audio only
Integrating Opus (2)

- Next step was integration in the core itself
  - Additional codec in conference bridge
  - Available since IETF86 in Orlando
- Open source implementation
  - New Opus codec module implemented for Asterisk 11
    - More on this in a minute…
- Made available for WebRTC remote attendees
  - Chrome (IETF86) and Firefox (IETF87)
  - Other endpoints not modified, all interoperable
    - Standards are nice!
Asterisk integration

- Asterisk integration made available as open source
- **Transcoding support for Asterisk 11**
  - [https://github.com/meetecho/asterisk-opus](https://github.com/meetecho/asterisk-opus)
    - Opus (transcoding) and VP8 (passthrough)
  - Automatically caps Opus to peer capabilities
    - e.g., Opus capped at 8kHz if talking to G.711
  - Needs work, but good feedback so far
- **Passthrough support for (upcoming) Asterisk 12**
  - [https://issues.asterisk.org/jira/browse/ASTERISK-21981](https://issues.asterisk.org/jira/browse/ASTERISK-21981)
    - Opus and VP8 (passthrough only)
    - Working with Asterisk community on this
Open Mike