Audio and Video Codecs for Real-Time Communications in the Browser

Koen Vos, Jean-Marc Valin, Timothy B. Terriberry

Introduction

For interoperability reasons, it is important to agree on standardized audio and video codecs in browser-based real-time communication (RTC) applications. This document presents a proposal for real-time audio and video coding that is royalty-free and addresses a broad range of applications. These codecs would be mandatory to implement in a browser order to to enable RTC compatibility with other browsers.

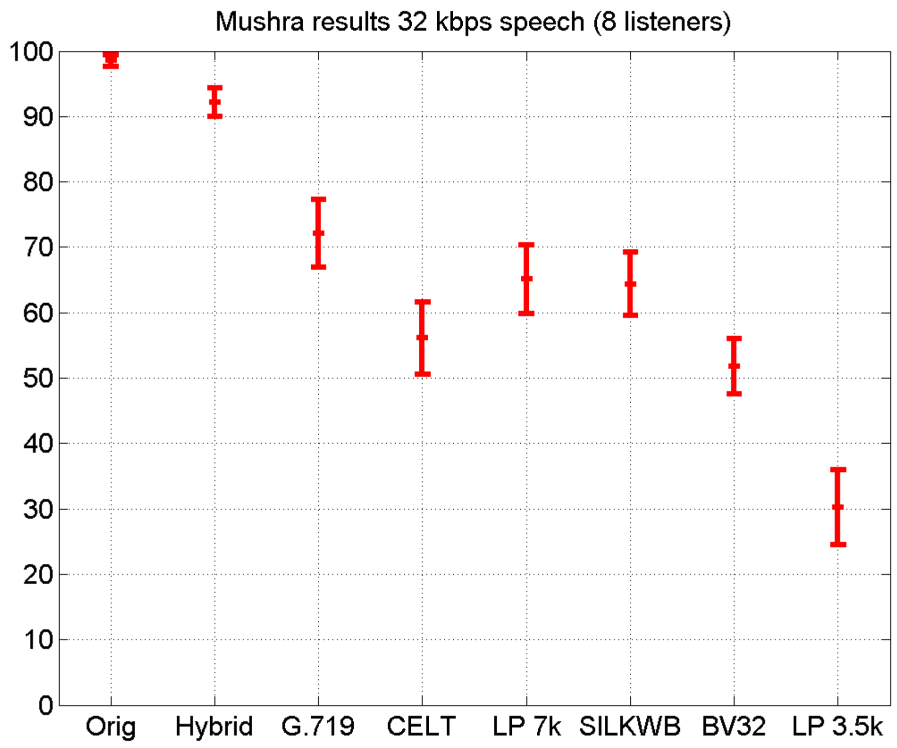
Audio: IETF codec

Work has been ongoing at the IETF since early 2009 to standardize [1] a speech/audio codec for transmission of interactive audio over the Internet. The technical requirements [2] are meant to cover a broad range of interactive audio applications on the Internet and include a wide range of audio bandwidths and bitrates. Another goal for the codec is to be "widely distributable", with no requirement to pay licensing fees of any kind.

The codec so far accepted as a working group item [3] is a hybrid between the SILK [4,5] and CELT [6,7] codecs, and has three modes:

- SILK only: optimized for low-bitrate voice up to 16 kHz sampling
- SILK + CELT: optimized for medium-low bitrate voice up to 48 kHz
- CELT only: optimized for medium-high bitrate audio up to 48 kHz

The supported bit-rates range from 6 kb/s for narrowband voice to 128+ kb/s per channel for perceptually lossless music. In practice, the SILK+CELT mode produces high quality full-band speech at 32 kbps, while the CELT-only mode produces high quality music at 64 kb/s per channel. There are multiple frame sizes available from 2.5 ms to 60 ms.
The mode, bit-rate, audio bandwidth, and frame size may all be changed on-the-fly using in-band signalling.
An informal MUSHRA listening test was conducted in July 2010 comparing the proposed codec to other codecs for a bit-rate of 32 kb/s. The results are presented in the figure below and clearly show that the proposed codec out-performs G.719, SILK alone, CELT alone, BV32, as well as uncompressed audio lowpass-filtered with cutoff frequencies of 7 or 3.5 kHz.



The codec’s source code [8] is available under the BSD license. Skype provides a free license for the patents included in the SILK codec and any IETF standard derived from it. No known patents cover CELT.
The timeline for this codec is to complete the outstanding changes by the next IETF meeting in Beijing in November, and have a release candidate by the end of 2010.

Video: VP8

VP8 was developed by On2 Technologies and recently open-sourced by Google as part of their WebM project to make royalty-free codecs available for use in HTML5 [9,10]. It is the most advanced of the royalty-free video formats, currently performing better than H.264 Baseline Profile and somewhat behind H.264 High Profile, though future encoder enhancements may help reduce that gap. Computational complexity is similar to that of H.264, while memory usage is comparable to the lowest levels of the H.264 profiles, because VP8 is limited to at most three reference frames. Although not currently a product of any standards body, the VP8 bitstream is frozen and documented, and Google is investigating avenues of standardization. There is currently no RTP specification for VP8, however: one will need to be developed.
Although the MPEG-LA has made generic statements indicating they believe it is likely that VP8 reads on patents owned by their members, It is safe to assume that Google did their legal homework before spending $126 million to acquire On2 [11], and the fact that they deploy VP8 on YouTube indicates they believe that it is safe enough for their own use. When Microsoft attempted to open WMV9, eventually standardized as VC-1 by the SMPTE, the MPEG-LA was able to quickly identify 12 companies (eventually expanded to 16) with rights to the technology. Despite much public posturing, no such concrete step has been taken against VP8. For these reasons, we believe the overall legal risk is low. It may still be prudent to develop a back-up plan in case of an eventual challenge to VP8's royalty-free terms, but at this point we assume one will

not be required.
VP8 was deployed by On2 Technologies for video-conferencing, and thus already has many of the features required for low-latency, interactive communication, including cyclic refresh and support for non-adaptive probability models and a tiered reference frame structure to improve tolerance to packet loss. VP8 also supports dynamic resolution switching and multiple "levels" which use reduced-complexity algorithms for lower-powered devices. The use of these features requires some knowledge of the target device or feedback to the encoder about the currently available CPU in the decoder. This will be a serious issue for mobile devices until hardware implementations of VP8 become common. In addition to all of these features, the existing open-source implementation, libvpx, would benefit from a number of enhancements.

The cyclic refresh feature is designed to distribute the bits required for a high-quality, static background over many frames to alleviate the need for large keyframes. However, for lossy transports such as UDP, it is also necessary to force macroblocks (MBs) to be coded in INTRA mode to guarantee recovery from packet loss without a keyframe, i.e., "rolling INTRA". This requires restricting the coding mode of some MBs, as well as restricting the motion vectors of all other MBs to ensure they do not reference "dirty" portions of the image after a refresh.
libvpx could also benefit from enhancements to its rate control algorithms. It currently uses frame-based rate control, where the target bitrate for a frame is used to select a single quantizer and rate-distortion multiplier, which is held constant throughout the entire frame. Frame-based rate control is computationally very efficient, but does not hit the desired rate target reliably on a per-frame basis, requiring a buffer multiple frames in size to absorb the variation in frame sizes. Such buffers are unusable for low-latency applications. In order to meet a maximum frame size target reliably enough to eliminate such a buffer, row-based or macroblock-based rate control is required.
In addition, the current rate control methods in libvpx allow setting a desired bitrate with hard limits on the quantizer used, which can occasionally result in rates which deviate greatly from the desired one. For low-latency applications where bandwidth is expensive, the opposite is needed: a desired quality with hard limits on the bitrate used. This allows the encoder to use fewer bits for simple content, saving money, while still ensuring it can fit within the channel capacity for difficult content, enabling low-latency.
The libvpx encoder allows the calling application to create a tiered reference structure, where frames at a given tier are only predicted from other frames at that tier or higher, with each higher tier being updated less frequently than the last. This allows complete recovery from a lost packet at one tier as soon as a frame from a higher tier is received. Currently libvpx requires the calling application to manually enforce this tiered structure. This is done by setting flags each time a frame is submitted for compression to control the type of frame produced and the list of reference frames it can use. This would benefit from integration into libvpx itself, due to interactions with other features such as rate control and rolling INTRA.
Finally, reference frame invalidation is a tool that allows a decoder to inform an encoder about lost packets, so that the encoder can produce a new frame using only reference frames the decoder is known to have received. Unlike audio, video can often recover from losses with this technique before a user even notices them. libvpx is already capable of this via the aforementioned flags to control the frame type and the valid references, but there is no existing open-source software using libvpx that implements this feature.

All told, these features would probably require a full-time engineer three to six months to implement and test. However, we do not think their lack is a serious obstacle to deploying VP8 sooner than that.

Other considerations

For best operation, the codecs need the following functionality from the browser:

- Audio/video transport statistics such as available bandwidth, packet loss, CPU utilization/ battery life. In turn, this requires a feedback channel;
- It may be desirable to implement a forward error correction (FEC) scheme for audio and video;
- Voice quality enhancement algorithms, such as acoustic echo cancellation (AEC), noise reduction and adaptive gain control (AGC);
- Interface for selecting whether the communication should be optimized for speech or music;
- Selection of the audio/video input devices;
- Control and feedback mechanism for controlling the CPU usage of video encoding and decoding.

Conclusion

We believe that the proposed combination of VP8 and the IETF audio codec are suitable for most, if not all, Internet real-time communication applications. We have shown where additional work on these codecs is needed or beneficial, and what supplemental functionality is required in the browser.

References

[1] IETF codec guidelines: http://tools.ietf.org/html/draft-valin-codec-guidelines-06
[2] IETF codec requirements: http://tools.ietf.org/html/draft-ietf-codec-requirements-01
[3] IETF codec description: http://tools.ietf.org/html/draft-ietf-codec-description-00
[4] SILK draft: http://tools.ietf.org/html/draft-vos-silk-01
[5] SILK website: http://developer.skype.com/silk
[6] CELT draft: http://tools.ietf.org/html/draft-valin-celt-codec-02
[7] CELT website: http://www.celt-codec.org/
[8] IETF codec git repository: git://git.xiph.org/users/jm/ietfcodec.git
[9] WebM project website: http://www.webmproject.org/
[10] VP8 git repository: git://review.webmproject.org/libvpx.git
[11] Analysis of WebM and its patent risk: http://carlodaffara.conecta.it/?p=420