Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2014

JM. Valin
T. Terriberry
Mozilla Corporation
K. Vos
Skype Technologies S.A.

July 12, 2013

Updates to the Opus Audio Codec
draft-valin-codec-opus-update-00

Abstract

This document addresses minor issues that were found in the specification of the Opus audio codec in RFC 6716 [RFC6716].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 13, 2014.

Copyright Notice

Table of Contents

1. Introduction

This document address minor issues that were discovered in the reference implementation of the Opus codec that serves as the specification in RFC 6716 [RFC6716]. Only issues affecting the decoder are listed here. An up-to-date implementation of the Opus encoder can be found at http://opus-codec.org/. The updated specification remains fully compatible with the original specification and only one of the changes results in any difference in the audio output.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Stereo State Reset in SILK

The reference implementation does not reinitialize the stereo state during a mode switch. The old stereo memory can produce a brief impulse (i.e. single sample) in the decoded audio. This can be fixed by changing silk/dec_API.c at line 72:

```
     for( n = 0; n < DECODER_NUM_CHANNELS; n++ ) {
         ret  = silk_init_decoder( &channel_state[ n ] );
     }
+    silk_memset(&((silk_decoder *)decState)->sStereo, 0,
+                sizeof(((silk_decoder *)decState)->sStereo));
+    /* Not strictly needed, but it's cleaner that way */
+    ((silk_decoder *)decState)->prev_decode_only_middle = 0;

     return ret;
 }
```

This change affects the normative part of the decoder. Fortunately, the modified decoder is still compliant with the original specification because it still easily passes the testvectors. For example, for the float decoder at 48 kHz, the opus_compare (arbitrary) "quality score" changes from from 99.9333% to 99.925%.

4. Parsing of the Opus Packet Padding

It was discovered that some invalid packets of very large size could trigger an out-of-bounds read in the Opus packet parsing code responsible for padding. This is due to an integer overflow if the signaled padding exceeds 2^31-1 bytes (the actual packet may be smaller). The code can be fixed by applying the following changes at line 596 of src/opus_decoder.c:

```
       /* Padding flag is bit 6 */
       if (ch&0x40)
       {
-         int padding=0;
          int p;
          do {
             if (len<=0)
                return OPUS_INVALID_PACKET;
             p = *data++;
             len--;
-            padding += p==255 ? 254: p;
+            len -= p==255 ? 254: p;
          } while (p==255);
-         len -= padding;
       }
```

This packet parsing issue is limited to reading memory up to about 60 kB beyond the compressed buffer. This can only be triggered by a compressed packet more than about 16 MB long, so it's not a problem for RTP. In theory, it _could_ crash a file decoder (e.g. Opus in Ogg) if the memory just after the incoming packet is out-of-range, but that could not be achieved when attempted in a production application built using an affected version of the Opus decoder.

5. Resampler buffer

The SILK resampler had the following issues:

1. The calls to memcpy() were using sizeof(opus_int32), but the type of the local buffer was opus_int16.

2. Because the size was wrong, this potentially allowed the source and destination regions of the memcpy overlap. We _believe_ that nSamplesIn is at least fs_in_khZ, which is at least 8. Since RESAMPLER_ORDER_FIR_12 is only 8,that should not be a problem once the type size is fixed.

3. The size of the buffer used RESAMPLER_MAX_BATCH_SIZE_IN, but the data stored in it was actually _twice_ the input batch size (nSamplesIn<<1).

The fact that the code never produced any error in testing (including when run under the Valgrind memory debugger), suggests that in practice the batch sizes are reasonable enough that none of the issues above was ever a problem. However, proving that is non-obvious.

The code can be fixed by applying the following changes to like 70 of silk/resampler_private_IIR_FIR.c:

```
        opus_int16                      out[],          /* O    Output
signal               */
        const opus_int16                in[],           /* I    Input
signal                */
        opus_int32                      inLen           /* I    Number of input
samples     */
    )
    {
        silk_resampler_state_struct *S = (silk_resampler_state_struct *)SS;
        opus_int32 nSamplesIn;
        opus_int32 max_index_Q16, index_increment_Q16;
   -    opus_int16 buf[ RESAMPLER_MAX_BATCH_SIZE_IN + RESAMPLER_ORDER_FIR_12 ];
   +    opus_int16 buf[ 2*RESAMPLER_MAX_BATCH_SIZE_IN +
RESAMPLER_ORDER_FIR_12 ];

        /* Copy buffered samples to start of buffer */
   -    silk_memcpy( buf, S->sFIR, RESAMPLER_ORDER_FIR_12 *
sizeof( opus_int32 ) );
   +    silk_memcpy( buf, S->sFIR, RESAMPLER_ORDER_FIR_12 *
sizeof( opus_int16 ) );

        /* Iterate over blocks of frameSizeIn input samples */
        index_increment_Q16 = S->invRatio_Q16;
        while( 1 ) {
            nSamplesIn = silk_min( inLen, S->batchSize );

            /* Upsample 2x */
            silk_resampler_private_up2_HQ( S->sIIR,
&buf[ RESAMPLER_ORDER_FIR_12 ], in, nSamplesIn );
```

```
            max_index_Q16 = silk_LSHIFT32( nSamplesIn, 16 + 1 );         /* + 1
because 2x upsampling */
            out = silk_resampler_private_IIR_FIR_INTERPOL( out, buf,
max_index_Q16, index_increment_Q16 );
            in += nSamplesIn;
            inLen -= nSamplesIn;

            if( inLen > 0 ) {
```

```
                /* More iterations to do; copy last part of filtered signal to
beginning of buffer */
   -            silk_memcpy( buf, &buf[ nSamplesIn << 1 ],
RESAMPLER_ORDER_FIR_12 * sizeof( opus_int32 ) );
   +            silk_memmove( buf, &buf[ nSamplesIn << 1 ],
RESAMPLER_ORDER_FIR_12 * sizeof( opus_int16 ) );
            } else {
                break;
            }
        }

        /* Copy last part of filtered signal to the state for the next call */
   -    silk_memcpy( S->sFIR, &buf[ nSamplesIn << 1 ], RESAMPLER_ORDER_FIR_12 *
sizeof( opus_int32 ) );
   +    silk_memcpy( S->sFIR, &buf[ nSamplesIn << 1 ], RESAMPLER_ORDER_FIR_12 *
sizeof( opus_int16 ) );
    }
```

6. Downmix to Mono

The last issue is not strictly a bug, but it is an issue that has been reported when downmixing Opus decoded stream to mono, whether this is done inside the decoder or as a post-processing on the stereo decoder output. Opus intensity stereo allows optionally coding the two channels 180-degrees out of phase on a per-band basis. This provides better stereo quality than forcing the two channels to be in phase, but when the output is downmixed to mono, the energy in the affected bands is cancelled sometimes resulting in audible artefacts.

A possible work-around for this issue would be to optionally allow the decoder to not apply the 180-degree phase shift when the output is meant to be downmixed (inside or outside of the decoder).

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Acknowledgements

We would like to thank Juri Aedla for reporting the issue with the parsing of the Opus padding.

9. References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6716] Valin, JM., Vos, K., and T. Terriberry, "Definition of the Opus Audio Codec", RFC 6716, September 2012.

Authors' Addresses

Jean-Marc Valin
Mozilla Corporation
650 Castro Street
Mountain View, CA 94041
USA

Phone: +1 650 903-0800
Email: jmvalin@jmvalin.ca

Timothy B. Terriberry
Mozilla Corporation
650 Castro Street
Mountain View, CA 94041
USA

Phone: +1 650 903-0800
Email: tterriberry@mozilla.com

Koen Vos
Skype Technologies S.A.
Soder Malarstrand 43
Stockholm 11825
SE

Phone: +46 73 085 7619
Email: koen.vos@skype.net