Abstract

This document proposes a mechanism for embedding very low bitrate deep audio redundancy (DRED) within the Opus codec (RFC6716) bitstream.

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 https://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 5 August 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
1. Introduction

This document proposes a mechanism for embedding very low bitrate deep audio redundancy (DRED) within the Opus codec [RFC6716] bitstream.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. DRED Description

Opus already includes a low-bitrate redundancy (LBRR) mechanism to transmit redundancy in-band to improve robustness to packet loss. LBRR is however limited to a single frame of redundancy, and typically uses about 2/3 of the bitrate of the "regular" Opus packet. The DRED extension allows up to one second or more [Open question: should we set a limit?] redundancy to be included in each packet, using a bitrate about 1/50 of the regular Opus bitrate.

DRED works by having the encoder transmit acoustic features in the Opus bitstream. On the receiver side, if packets are lost, then the first packet to arrive will contain the acoustic features for a certain duration in the past. The decoder can then use the features to synthesize the missing speech -- either from the last received or
from the last audio samples produced by packet loss concealment (PLC). Although the synthesized speech samples should be consistent with the last known samples at the point of the transition, the features do not contain waveform-specific or phase-specific information so the synthesized speech waveform will significantly deviate from the original waveform, despite sounding similar.

2.1. Acoustic Features

DRED uses 20 acoustic features to synthesize speech. The first 18 are Bark-frequency cepstral coefficients (BFCC) and the last represent the pitch frequency and the voicing information. The BFCC features are based on bands that match the CELT bands, as shown in Table 1.

<table>
<thead>
<tr>
<th>Band</th>
<th>Start frequency (Hz)</th>
<th>Center frequency (Hz)</th>
<th>End frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>400</td>
<td>600</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>600</td>
<td>800</td>
</tr>
<tr>
<td>4</td>
<td>600</td>
<td>800</td>
<td>1000</td>
</tr>
<tr>
<td>5</td>
<td>800</td>
<td>1000</td>
<td>1200</td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>1200</td>
<td>1400</td>
</tr>
<tr>
<td>7</td>
<td>1200</td>
<td>1400</td>
<td>1600</td>
</tr>
<tr>
<td>8</td>
<td>1400</td>
<td>1600</td>
<td>2000</td>
</tr>
<tr>
<td>9</td>
<td>1600</td>
<td>2000</td>
<td>2400</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2400</td>
<td>2800</td>
</tr>
<tr>
<td>11</td>
<td>2400</td>
<td>2800</td>
<td>3200</td>
</tr>
<tr>
<td>12</td>
<td>2800</td>
<td>3200</td>
<td>4000</td>
</tr>
<tr>
<td>13</td>
<td>3200</td>
<td>4000</td>
<td>4800</td>
</tr>
<tr>
<td>14</td>
<td>4000</td>
<td>4800</td>
<td>5600</td>
</tr>
<tr>
<td>15</td>
<td>4800</td>
<td>5600</td>
<td>6800</td>
</tr>
<tr>
<td>16</td>
<td>5600</td>
<td>6800</td>
<td>8000</td>
</tr>
<tr>
<td>17</td>
<td>6800</td>
<td>8000</td>
<td>8000</td>
</tr>
</tbody>
</table>

Table 1: Band definitions for DRED

TODO: Specify exact computation of the cepstral features and voicing. Open question: how do we specify the neural pitch estimator?

2.2. Rate-Distortion-Optimized Variational Autoencoder (RDO)

The features described above need to be transmitted to the decoder with the fewest number of bits possible. Although it is not acceptable to make redundancy from one packet depend on the redundancy of another packet, we can use as much prediction as we like within one packet. In practical use, the same audio feature
vector is included in many different packets (50 for 1 second redundancy). For that reason, we do not want to fully re-encode acoustic features for each packet. On the decoder side, since the most recent audio is the most likely to be used, we minimize the computation time by having the audio encoded from the most recent, going backward in time.

TODO: Specify the cepstral features and voicing. Open question: how do we specify the neural pitch estimator?

Figure 1: DRED encoding/decoding
2.2.1. Encoder architecture

Every 20 ms, the encoder takes in a pair of 20-dimensional acoustic feature vectors as input and produces one initial state (IS) and one latent vector. Each latent vector encodes 40 ms (their information overlaps), so only half the latent vectors need to be transmitted. Although an encoder is provided for reference, the encoder architecture is not normative. Each redundancy packet contains the latest initial state, along with latent vectors ordered from the latest (the one aligned with the initial state) to the earliest one the encoder includes. Each component of the IS and latent vectors are quantized and then entropy-coded following a Laplace distribution. The same procedure is used for both the latent vectors and the initial state (we will describe the process for a latent variable). The quantized index \(X\) is obtained by scaling the \(i\)'th latent variable \(z_i\) by a scaling factor \(s_{i,q}\) that depends on both \(i\) and on the quantizer \(q\). We then apply a "dead-zone" function

\[
\zeta(z) = z - d\tanh(z / (d + \text{epsilon})),
\]

where \(d\) also depends on \(i\) and \(q\), and \(\text{epsilon}=0.1\). The result is then rounded to the nearest integer: \(X = \text{round}(\zeta(s_{i,q}z_i))\). The Laplace distribution used for entropy coding is parameterized with a probability that the value is zero \((p_0)\), as well as a decay factor \(r\) \((0 < r < 1)\). Both \(p_0\) and \(r\) depend on \(i\) and \(q\). The probability \(p(X)\) for a coefficient is given by:

\[
P(X) = \begin{cases} 
  p_0, & \text{if } X = 0 \\
  (1 - p_0) \cdot r, & \text{if } X \neq 0 \\
  \frac{(1 - p_0) \cdot r}{2 \cdot (1 - r)}, & \text{if } X \neq 0
\end{cases}
\]

2.2.2. Decoder architecture

Unlike the encoder, the decoder is normative. The decoder uses the same Laplace distribution above to decode the symbols and then scales them back by \(1/s_{i,q}\). The initial state is used as input to initialize the decoder's gated recurrent units (GRUs). The latent vectors are used on at a time as input the DNN decoder, which produces 4 vectors of 20 acoustic features for each input latent vector.

Open question: how do we specify the decoder DNN architecture and (especially) the weights? We expect about 500k to 1M weights, most of which can be represented as 8-bit integers, the others as floating-point.
2.2.3. Statistical data

We define 16 different quantization settings, ranging from q=0 (higher bitrate) to q=15 (lower bitrate). For each quantizer and for each latent variable or initial state coefficient, we have a normative scale (s), decay (r), and p0 value. Note that the dead-zone parameters d are not normative.

*Insert tables with statistical model here*

2.2.4. Vocoder

A vocoder is needed to turn the acoustic features into actual speech to fill in the audio for any missing packets. Although the decoder is not normative, certain properties are needed for DRED to function adequately. First, the vocoder SHOULD be able to start synthesizing speech by continuing an existing waveform, reducing the artifacts caused at the beginning of a lost packet. If such property cannot be achieved, then the implementation SHOULD at least make an attempt to synchronize the phase of the synthesized speech with the last received speech, and attempt some form of blending, e.g. by splicing the signals in the LPC residual domain.

A second important property of the vocoder is to not rely on more than one feature vector of look-ahead. To synthesize speech between time t-10ms and t, the vocoder SHOULD NOT rely on acoustic features centered beyond t+5ms (i.e. covering t-5ms to t+15ms). The vocoder MAY use more look-ahead when it is available, but there are cases (e.g. last lost packet) where the amount of acoustic feature vectors will be limited. For frames sizes less than 20 ms, the decoder SHOULD be prelated to deal with having less than one feature vector of look-ahead.

3. DRED Extension Format

We use the Opus extension mechanism [opus-extension] to add deep redundancy within the padding of an Opus packet. We use the extension ID 32, which means that the L flag signals whether a length code is included. In this document, we define only the extension payload. [Note: until adoption by the IETF, experimental implementations of DRED MUST use experiment extension ID 126 to avoid causing interoperability problems]

The principles behind the DRED mechanism defined in this extension are explained in [dred-paper]. All the data in the extension payload is encoded using the Opus entropy coder defined in Section 4.1 of [RFC6716]. Since some of the fields at the beginning of the payload are encoded with flat binary probabilities, they can still be interpreted as bits.
The extension starts with an offset indicator, encoded as a signed 5-bit integer (two's complement) in units of 2.5 ms. The offset indicates the time of the last sample analysed for the transmitted features in the packet, measured from the time of the first sample in the Opus frame that contains the extension data.

The offset is followed by a 4-bit initial quantizer field (Q0) ranging from 0 to 15. That quantizer is used on the most recent frame encoded and is followed by the 3-bit quantizer slope dQ. The 3-bit dQ index selects from the following values: [0, 1/8, 3/16, 1/4, 3/8, 1/2, 3/4, 1] quantizer step per frame. The quantizer for frame k is thus given by: min(15, round(Q0 + dQ_table[dQ] * k)). For example, using Q0=5 and dQ=2 (3/16), frame k=20 would use a quantizer of round(5 + 3/16 * k) = 9.

The compressed redundancy information consists of an initial state coded with a pyramid vector quantizer (PVQ), followed by the entropy-coded latent representation. The number of 40-ms DRED blocks is not coded explicitly. Instead, the decoder MUST NOT decode blocks when fewer than 8 bits remain in the DRED payload.

```
+-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
| Offset | Q0   | dQ | Initial state vector               | +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
| 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 |        |                             +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
|       |      |    | +---------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
|       |      |    |   ...                             |  Latent vectors 0,                       |                             +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
|       |      |    |       +---------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
|       |      |    | Latent vector 1, ...              |                             +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
|       |      |    |       +---------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
|       |      |    | Latent vector n-1                 |  unused                              |                             +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
|       |      |    |       +---------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+  +-------------------------------------------+
```

Figure 2: Extension framing

4. IANA Considerations

[Note: Until the IANA performs the actions described below, implementers should use 126 instead of 32 as the extension number.]

This document assigns ID 32 to the "Opus Extension IDs" registry created in [opus-extension] to implement the proposed DRED extension.
4.1. Opus Media Type Update

This document updates the audio/opus media type registration [RFC7587] to add the following two optional parameters:

ext32-dred-duration: Specifies the maximum amount of DRED information (in milliseconds) that the receiver can use. The receiver MUST be able to handle any valid DRED duration even if it does not make use of it. The sender MUST NOT send more than the specified amount of redundancy to avoid leaking information beyond what the receiver expects.

sprop-ext32-dred-duration: Maximum amount of DRED information (in milliseconds) that the sender is likely to use. The receiver MUST be able to handle any valid DRED duration even if it does not make use of it. The sender MUST NOT send more than the specified amount of redundancy to avoid leaking information beyond what the receiver expects.

4.2. Mapping to SDP Parameters

The media type parameters described above map to declarative SDP and SDP offer-answer in the same way as other optional parameters in [RFC7587]. Regardless of any a=fmtp SDP attribute specified, the receiver MUST be capable of receiving any signal.

5. Security Considerations

When using a Selective Forwarding Unit (SFU), it is possible for the DRED payload to include speech that would not otherwise have been transmitted. For example, a new user joining may receive audio that was transmitted before them joining. If such behavior is a security or confidentiality concern, then the SFU SHOULD use the ext32-dred-duration and sprop-ext32-dred-duration parameters to limit the amount of redundancy and/or temporarily drop DRED payloads when that could leak information.

As is the case for any media codec, the decoder must be robust against malicious payloads. Similarly, the encoder must also be robust to malicious audio input since the encoder input can often be controlled by an attacker. That can happen through browser JS, echo, or when the encoder is on a gateway.

DRED is designed to have a complexity that is independent of the signal characteristics. However, there exist implementation details that can cause signal-dependent complexity changes. One example is CPU treatment of denormals that can sometimes cause increased CPU load and could be triggered by malicious input. For that reason, it is important to minimize such impact to reduce the impact of DOS attacks. Similarly, since the encoding and decoding process can be
computationally costly, devices must manage the complexity to avoid attacks that could trigger too much DRED encoding or decoding to be performed.

The use of variable-bitrate (VBR) encoding in DRED poses a theoretical information leak threat [RFC6562], but that threat is believed to be significantly lower than that posed by VBR encoding in the main Opus payload. Since this document provides a way to dynamically vary the amount of redundancy transmitted, it is also possible to reduce the overall VBR risk of Opus by using DRED as a way of making the total Opus payload constant (CBR) or nearly constant.

6. References

6.1. Normative References


6.2. Informative References


[dred-paper] Valin, J.-M., Buethe, J., and A. Mustafa, "Low-Bitrate Redundancy Coding of Speech Using a Rate-Distortion-

Authors' Addresses

Jean-Marc Valin
Xiph.Org Foundation
Canada

Email: jmvalin@jmvalin.ca

Jan Buethe
Amazon
Germany

Email: jbuethe@amazon.com