Internet Engineering Task Force
Internet-Draft
Updates: 6716 (if approved)
Intended status: Standards Track
Expires: 8 January 2026
7 July 2025

JM. Valin
Google
J. Buethe
Xiph.Org Foundation

Deep Audio Redundancy (DRED) Extension for the Opus Codec

draft-ietf-mlcodec-opus-dred-04

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

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 2
2. DRED Description 2
2.1. Acoustic Features 3
2.2. Rate-Distortion-Optimized Variational Autoencoder (RDO) 5
2.2.1. Encoder architecture 6
2.2.2. Decoder architecture 7
2.2.3. Statistical data 10
2.2.4. Vocoder 19
3. DRED Extension Format 19
3.1. Latent decoding 21
4. Conformance 22
4.1. DRED Feature Decoding 22
4.2. Vocoder 22
5. IANA Considerations 23
5.1. Opus Media Type Update 23
5.2. Mapping to SDP Parameters 23
6. Security Considerations 23
7. References 24
7.1. Normative References 24
7.2. Informative References 25
Authors' Addresses 25

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 redundancy to be included in each packet, using a bitrate about 1/50 of the regular

Opus bitrate. Although the amount of redundancy that can be encoded in a packet is unbounded, there appears to be little use to including more than a few seconds.

DRED is transmitted within the Opus padding, as described in [opus-extension]. In the case of multi-frame packets, there SHOULD only be one DRED extension per packet and it SHOULD be associated with the first frame of the packet. In all cases, there MUST NOT be more than one DRED extension associated to the same frame.

The DRED encoder SHOULD remove any leading or trailing silence from the redundant audio data. That being said, silence that occurs between speech segments cannot be left out. Any Selective Forwarding Unit (SFU) designed not to forward silent packets SHOULD still forward DRED-containing packets from the last known active source. Conference mixers SHOULD either forward DRED from the last known active source or re-encode DRED from the mixed audio.

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.

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

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?

```
                                 Audio
                                   |
                                   v
                           +---------------+
                           | RDOVAE encoder|
                           +---------------+
                                   |
                                   v
       +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
       | L | L | L | L | L | L | L | L | L | L | L | L | L | L |
       +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
       | S | S | S | S | S | S | S | S | S | S | S | S | S | S |
       +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                                         |   |   |
                                         v   |   |
               +---+---+---+---+---+---+---+ |   |
    decoder <--| L |   | L |   | L |   | L | |   |
               +---+---+---+---+---+---+---+ |   |
                                       | S | |   |
                                       +---+ |   |
                                             v   |
                   +---+---+---+---+---+---+---+ |
        decoder <--| L |   | L |   | L |   | L | |
                   +---+---+---+---+---+---+---+ |
                                           | S | |
                                           +---+ |
                                                 v
                       +---+---+---+---+---+---+---+
            decoder <--| L |   | L |   | L |   | L |
                       +---+---+---+---+---+---+---+
                                               | S |
                                               +---+
```

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 conponent 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 $zeta(z) = z - d*tanh(z / (d + epsilon))$, where d also depends on i and q, and epsilon=0.1. The result is then rounded to the nearest integer: $X = round(zeta(s_{i,q}*z_i))$. The Laplace distribution used for entropy coding is parameterized with a probability that the value is zero (p0), as well as a decay factor r ($0 < r < 1$). Both p0 and r depend on i and q. The probability p(X) for a coefficient is given by:

$$
P(X) = \begin{cases} p0 & , \quad \text{if } X = 0 \\ \dfrac{(1 - p0) * r^{|X|}}{2 * (1 - r)} & , \quad \text{if } X \neq 0 \end{cases}
$$

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 one at a time as input the DNN decoder, which produces 4 vectors of 20 acoustic features for each input latent vector.

The decoder is mostly structured as a DenseNet network, with 5 sets of alternating GRU and convolutional layers. Let gru1..gru5 denote the 5 GRUs, conv1..conv5 denote the 5 convolutional layers, hidden_init/gru_init/dense1/output denote fully-connected layers, glu1..glu5 denote gated linear units (GLUs), and cat() denote tensor concatenation. All GRU layers have 96 outputs (number of neurons) and all convolutional layers have 32 outputs. Despite using a functional notation, both the GRU and convolutional layers have an internal state when used one latent vector at a time. The fully-connected layers all have different sizes. Unless otherwise noted, the GRUs, convolutional and fully-connected layers all use tanh output activations and the GRUs use sigmoid as gate activation. GLUs are defined as:

$$L(y) = sigmoid(W*y)*y$$

where y is the input and W is a square matrix of the same dimensions as y. The decoder starts with the 18-dimensional initial state vector IS. The IS is used to compute the GRU initialization vector V using both hidden_init and gru_init:

```
V = gru_init(hidden_init(IS))
```

where hidden_init has 18 inputs and 128 output, and D2 has 128 inputs and 480 (5*96) outputs. The components of V are split (sequentially) into the V1..V5 initialization vectors (original state before the decoding process) for GRUs gru1..gru5. Let Z be the decoded 20-dimensional latent vector for a particular 40-ms chunk. From there, the DenseNet structure can be expressed as:

```
t1 = dense1(Z)
t2 = cat(t1, conv1(t1))
t3 = cat(t2, glu1(gru1(t2)))
t4 = cat(t3, conv2(t3))
t5 = cat(t4, glu2(gru2(t4)))
t6 = cat(t5, conv3(t5))
t7 = cat(t6, glu3(gru3(t6)))
t8 = cat(t7, conv4(t7))
t9 = cat(t8, glu4(gru4(t8)))
t10 = cat(t9, conv5(t9))
t11 = cat(t10, glu5(gru5(t10)))
x = output(t11)
```

where t1..tN are temporary vectors and "output" is the only layer to have a linear output activation, with 80 output neurons (4*20). The dimensionality of t1..t11 (and corresponding GRU/convolutional input size) can be inferred from the concatenation operations. The output vector x is split (sequentially) into 4 feature vectors of 20 dimensions each that can be sent to the vocoder is packets are lost.

2.2.2.1. Decoder weights

The decoder weights are distributed outside of this document at https://media.xiph.org/opus/ietf/draft-ietf-mlcodec-opus-dred-01-weights.bin. [FIXME: Find permanent location for the weights] They are distributed in a simple binary format that can also be used to separate them from an implementation binary for easier downloads. Each weight matrix is stored separately as a single array block. Each block starts with a 64-byte header, followed by a multiple of 64 bytes of array data. Blocks are self-delimited and can be concatenated into a single file.

The header starts with a 4-byte Header ID representing the string "DNNw", followed by a 4-byte Version number (currently 0). The Type of the weights follows, encoded as a 4-byte integer, where value 0 represents floating point weights and value 3 represents 8-bit signed integers. The 4-byte Size field that follows represents the size of the data in bytes (not number of elements), and the Block Size is the number of data bytes rounded up to 64 bytes. The block size indicates where the next block is expected. Note that the block size does not include the header size. The remaining 44 bytes of the header contain the name of the array.

For implementation efficiency, the binary format can be implemented using any endianness, but for the purpose of distributing the reference weights, we use a little-endian format.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Header ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Version                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Type                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Size                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Block size                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Name (44 bytes)                       |
:                              ...                              :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Data (N x 64 bytes)                     |
:                              ...                              :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Binary Weights Format

The decoder arrays are named dec_<layer name>_<variable name>, where the names are gru1..gru5, con1..conv5, and so on. There is an optional _float or _int8 suffix for type when relevant. Variable names can be "bias", "subias", "scale" and "weights". TODO: more on how the matrices are used.

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.

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	255	208	168	134	106	82	64	48	36	26	17	10	3	3	2	2
1	255	219	187	160	137	117	101	81	70	50	23	6	6	5	3	2
2	255	218	187	160	138	118	102	84	71	63	31	7	7	5	3	2
3	255	217	186	159	137	118	102	87	76	66	53	25	11	5	2	1
4	255	216	183	155	131	111	95	79	67	57	48	42	35	29	24	21
5	255	219	189	163	141	122	107	90	87	31	11	3	3	2	1	1
6	255	218	187	160	138	119	103	87	72	45	18	6	5	3	2	2
7	255	217	184	157	133	113	96	78	67	53	34	17	6	5	4	3
8	255	222	192	167	146	128	114	87	78	63	40	9	8	6	4	3
9	255	217	184	157	135	115	99	84	73	65	56	48	18	11	6	2
10	255	219	189	163	141	122	107	90	74	40	15	5	4	3	2	1
11	255	214	180	151	127	108	91	76	65	56	47	41	35	31	27	24
12	255	215	181	152	129	109	93	78	67	57	49	43	38	33	29	27
13	255	218	187	160	138	119	102	87	75	56	34	19	7	4	2	2
14	255	219	188	162	139	120	103	80	69	34	12	3	3	2	1	1
15	255	219	189	164	143	124	108	69	20	5	0	1	1	1	1	0
16	255	217	185	158	136	117	101	86	76	67	58	47	15	11	7	6
17	255	217	184	157	135	115	99	84	74	63	54	47	16	10	7	5
18	255	213	178	149	124	104	87	72	60	50	42	35	29	25	21	18
19	255	215	181	152	127	105	86	58	46	21	10	2	0	0	0	0
20	255	214	179	149	125	104	87	72	61	51	43	36	31	27	23	20

Table 2: Scale values for latent

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	1	1	0	0	0	1	1	2	3	12	27	44	178	255	255	255
1	0	0	7	17	29	45	70	107	160	255	255	255	255	255	255	255
2	10	13	16	20	24	29	35	41	53	255	255	255	255	255	255	255
3	0	1	5	9	14	20	26	37	51	81	124	255	255	255	255	255
4	0	0	0	1	4	6	9	11	16	24	37	53	87	108	255	255
5	6	12	17	24	31	41	56	85	255	255	255	255	255	255	255	255
6	11	15	18	22	27	33	41	48	53	255	255	255	255	255	255	255
7	0	0	0	5	11	17	27	46	75	124	220	255	255	255	255	255
8	0	8	25	43	66	94	133	168	231	255	255	255	255	255	255	255
9	0	0	2	6	11	16	23	31	44	71	104	158	255	255	255	255
10	7	12	17	22	28	36	47	59	81	255	255	255	255	255	255	255
11	0	0	0	1	2	4	5	7	9	12	15	19	23	27	30	38
12	0	0	1	2	4	6	9	11	14	20	28	37	57	65	75	96
13	0	3	7	11	16	21	28	39	54	67	255	255	255	255	255	255
14	13	18	22	28	34	43	56	72	255	255	255	255	255	255	255	255
15	0	0	4	13	23	37	56	255	255	255	255	255	255	255	255	255
16	4	7	11	14	19	24	30	39	49	70	96	123	255	255	255	255
17	0	0	3	7	11	16	21	28	38	54	73	108	255	255	255	255
18	0	0	0	0	0	0	0	0	0	0	2	3	5	7	9	11
19	5	12	18	26	34	43	56	84	255	255	255	255	255	255	255	255
20	0	0	0	0	0	0	1	2	3	5	8	11	14	16	18	21

Table 3: Dead zone values for latent

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	233	228	222	214	204	191	176	155	135	106	66	32	0	0	0	0
1	94	85	72	59	45	32	21	10	4	0	0	0	0	0	0	0
2	91	75	58	43	29	17	9	4	2	0	0	0	0	0	0	0
3	112	96	81	65	51	38	26	16	10	4	1	0	0	0	0	0
4	149	138	125	109	93	77	61	45	32	21	12	7	3	1	0	0
5	65	50	36	24	14	8	4	2	0	0	0	0	0	0	0	0
6	92	75	59	43	29	18	10	5	2	0	0	0	0	0	0	0
7	118	107	97	74	60	48	38	29	17	6	0	0	0	0	0	0
8	55	47	36	27	19	13	8	3	2	0	0	0	0	0	0	0
9	122	107	92	76	60	46	34	22	15	9	4	2	0	0	0	0
10	82	67	53	40	29	20	14	8	4	0	0	0	0	0	0	0
11	190	181	171	160	149	135	120	101	85	68	52	38	26	17	10	6
12	175	165	154	143	128	113	98	81	67	53	41	31	23	15	9	5
13	100	85	70	56	42	31	21	12	6	1	0	0	0	0	0	0
14	80	64	49	35	23	14	7	2	0	0	0	0	0	0	0	0
15	62	47	33	21	12	6	3	0	0	0	0	0	0	0	0	0
16	125	109	92	75	59	43	30	18	10	5	1	1	0	0	0	0
17	130	114	98	82	66	50	37	24	15	7	2	1	0	0	0	0
18	236	233	229	224	219	213	206	198	189	180	169	158	146	132	118	104
19	90	72	54	37	24	15	9	3	0	0	0	0	0	0	0	0
20	219	213	207	199	190	181	172	160	148	133	118	103	88	74	62	51

Table 4: Decay (r) values for latent

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	12	14	18	22	27	35	44	57	78	106	152	201	255	255	255	255
1	162	171	184	197	211	224	235	246	252	255	255	255	255	255	255	255
2	137	147	158	171	184	198	212	228	241	255	255	255	255	255	255	255
3	134	142	152	163	175	188	201	216	228	242	253	255	255	255	255	255
4	107	118	126	135	144	155	166	179	192	207	223	235	248	253	255	255
5	138	152	167	183	199	215	231	246	255	255	255	255	255	255	255	255
6	118	130	144	158	174	190	206	223	237	255	255	255	255	255	255	255
7	138	149	159	167	180	194	208	227	239	250	255	255	255	255	255	255
8	201	209	220	229	237	243	248	253	254	255	255	255	255	255	255	255
9	114	123	133	145	158	172	186	204	218	234	246	253	255	255	255	255
10	145	157	169	182	196	209	223	237	248	255	255	255	255	255	255	255
11	66	75	85	96	107	115	122	132	140	151	163	175	189	201	213	224
12	81	91	102	113	122	131	140	153	164	177	192	205	220	230	238	244
13	143	153	163	175	187	199	211	226	237	249	255	255	255	255	255	255
14	146	157	170	183	198	213	228	245	255	255	255	255	255	255	255	255
15	159	168	179	193	208	222	237	255	255	255	255	255	255	255	255	255
16	122	130	140	150	161	174	187	203	216	232	245	253	255	255	255	255
17	121	128	137	147	159	170	183	198	212	228	241	250	255	255	255	255
18	20	23	27	32	37	43	50	58	67	76	87	98	108	116	125	134
19	104	120	139	159	182	205	227	251	255	255	255	255	255	255	255	255
20	37	43	49	57	66	75	84	96	106	115	126	137	148	159	169	180

Table 5: P(0) values for latent

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	255	215	181	153	129	109	93	78	67	58	51	45	40	35	31	27
1	255	215	181	153	128	108	91	77	65	55	47	41	36	31	27	24
2	255	233	205	175	146	120	97	77	62	49	40	33	27	23	19	15
3	255	215	181	152	127	107	89	74	62	53	44	37	32	28	24	21
4	255	216	182	154	131	111	95	81	70	63	57	51	47	41	36	31
5	255	215	181	152	128	108	91	76	64	55	46	39	34	29	25	21
6	255	216	182	155	131	111	95	81	71	65	60	53	47	41	36	32
7	255	216	183	155	132	113	98	87	79	79	78	69	62	53	46	40
8	255	215	181	152	128	108	91	77	65	56	47	41	36	31	27	24
9	255	216	183	155	131	112	96	82	71	62	54	47	41	37	34	42
10	121	114	102	84	61	43	31	1	0	2	131	188	255	216	181	151
11	255	215	182	153	129	108	91	77	65	55	47	40	34	28	24	20
12	255	217	184	155	130	110	92	77	64	54	45	38	32	27	23	19
13	255	227	196	166	140	118	98	82	69	57	48	40	34	29	24	20
14	255	216	182	154	130	110	93	80	69	60	53	47	42	37	32	28
15	255	216	184	156	133	114	98	87	77	72	66	59	52	46	40	36
16	255	216	184	156	134	115	100	91	82	77	67	59	52	46	40	36
17	255	216	183	155	131	110	93	78	66	57	49	42	37	32	28	25
18	71	65	60	54	49	45	42	45	49	92	189	235	255	213	177	146

Table 6: Scale values for state

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	13	12	11	11	11	11	11	11	11	13	12	9	7	13	19	26
1	16	14	12	11	9	8	7	4	4	4	4	5	7	5	3	7
2	9	8	7	6	6	4	3	3	2	3	2	0	3	2	4	4
3	6	8	8	9	9	9	10	8	8	11	11	10	15	22	28	37
4	20	18	17	16	15	15	15	14	13	14	13	9	9	14	21	30
5	10	8	7	5	4	4	3	3	2	3	4	6	8	9	10	10
6	13	13	13	13	13	13	14	12	12	11	2	1	10	17	24	34
7	35	30	25	22	19	17	16	18	15	22	0	1	0	4	7	12
8	13	11	9	8	6	5	4	3	2	3	3	4	9	6	2	5
9	15	15	15	15	15	16	17	17	18	16	20	26	34	46	75	255
10	255	255	255	255	255	255	255	255	255	2	0	0	0	0	0	0
11	9	7	6	5	4	3	2	1	1	0	0	1	2	2	3	3
12	11	9	6	5	3	2	2	2	2	3	4	4	3	3	3	2
13	10	8	6	5	4	3	2	2	1	2	2	2	4	3	4	1
14	23	19	17	14	12	11	9	8	8	11	9	4	4	7	9	13
15	14	14	14	15	16	17	18	20	18	0	8	13	14	23	33	50
16	26	24	21	19	17	16	12	7	0	11	14	14	17	24	32	46
17	43	38	32	27	22	18	14	7	1	0	0	0	0	0	0	0
18	255	255	255	255	255	255	255	255	255	121	29	4	1	0	1	4

Table 7: Dead zone values for state

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	207	199	190	181	169	158	145	130	116	103	90	77	66	52	39	27
1	224	218	212	205	196	187	177	165	152	139	126	112	101	87	74	60
2	253	253	252	252	251	250	249	247	245	242	239	235	231	226	220	213
3	207	199	190	180	169	157	144	128	113	99	82	68	56	46	37	30
4	197	187	177	165	152	139	124	109	95	84	74	64	56	42	30	19
5	233	229	224	218	212	205	197	187	177	166	154	140	127	112	97	81
6	190	181	170	158	144	130	115	100	86	78	70	60	48	36	25	16
7	198	189	178	167	154	141	127	115	106	107	107	96	86	71	57	43
8	232	227	223	217	210	203	194	183	173	161	149	136	124	111	99	84
9	180	168	156	143	128	112	97	79	64	50	37	25	17	10	7	5
10	4	3	1	0	0	0	0	0	0	0	19	104	132	117	100	83
11	245	243	240	237	234	230	226	220	214	208	200	191	182	171	160	147
12	251	251	250	249	247	246	244	241	239	235	232	227	222	216	210	202
13	254	253	253	253	252	251	250	249	248	246	244	242	239	236	233	229
14	210	203	194	185	174	162	149	136	122	109	98	88	78	64	51	38
15	173	162	149	135	120	105	91	78	67	63	53	43	32	22	15	9
16	169	156	142	128	112	98	85	77	71	61	48	37	28	18	10	5
17	223	218	212	205	197	188	179	166	155	143	131	120	110	99	89	79
18	22	17	12	7	4	2	1	2	11	90	166	183	188	178	164	150

Table 8: Decay (r) values for state

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	40	45	52	59	67	75	84	95	105	115	124	132	139	153	167	182
1	24	28	32	37	43	49	56	63	72	80	90	100	110	119	128	142
2	1	2	2	2	2	3	4	5	6	7	9	11	13	16	19	23
3	35	41	48	56	65	75	85	97	109	124	139	153	168	183	197	210
4	45	50	56	64	72	81	90	101	110	118	125	132	139	155	171	188
5	15	18	21	24	29	33	39	45	52	60	69	78	88	98	108	119
6	47	54	62	70	79	89	99	110	119	126	127	136	150	167	183	200
7	44	49	54	60	67	74	82	90	95	97	91	99	107	121	135	151
8	15	17	20	23	27	31	35	40	46	53	61	70	78	88	96	109
9	58	65	73	82	92	102	112	125	136	146	160	176	193	209	226	251
10	252	253	255	255	255	255	255	255	255	255	189	93	72	83	96	110
11	7	8	9	11	13	15	18	21	24	29	33	39	46	53	60	69
12	2	3	3	4	4	5	6	7	9	11	13	15	17	21	24	29
13	1	1	1	2	2	2	3	4	4	5	6	7	8	10	12	14
14	25	28	33	39	45	52	60	70	79	89	98	106	114	128	142	157
15	56	64	73	83	93	105	116	128	135	131	142	155	168	185	201	218
16	53	61	69	78	88	98	109	116	121	131	145	159	172	188	204	220
17	17	21	25	31	39	45	52	58	65	74	84	94	105	116	128	139
18	230	235	240	246	250	252	254	251	235	129	50	39	36	43	51	60

Table 9: P(0) values for state

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 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: q=min(Qmax, round(Q0 + dQ_table[dQ] * k)), where Qmax is the maximum quantizer allowed. For example, using Q0=5 and dQ=2 (3/16), frame k=20 would use a quantizer of round(5 + 3/16 * k) = 9.

We then have one bit (X) that flags whether an extended offset is used. If X=0, then a 5-bit offset indicator follows. The offset is a positive integer in units of 2.5 ms. It indicates the time of the last sample analysed for the transmitted features in the packet, measured from 40ms after the first sample in the Opus frame that contains the extension data.

If X=1, then we have an extended offset field, with an additional 8 bits to signal the offset. This makes it possible to signal a maximum offset of (2^13-1)*2.5ms, or approximately 20.5 seconds.

If Q0<14 and dQ!=0, then the offset is followed by the range-coded Qmax parameter. The probability of Qmax=15 is set to 1/2 (one bit is used), whereas other possible values (Q0 < Qmax < 15) are coded with a flat probability distribution. The pdf for Qmax is {nval, 1, 1, ...}/(2*nval), where there are nval=14-Q0 ones. The Qmax=15 symbol is first, followed by other values in ascending order, starting from Qmax=Q0+1.

The compressed redundancy information consists of an initial state coded, followed by a sequence of 40-ms latent vectors. Both the initial state and the latent vectors are entropy-coded using a Laplace distribution. The number of 40-ms DRED latent vectors is not coded explicitly. Instead, the decoder keeps decoding them until it runs out of bits. More specifically, the decoder MUST NOT decode blocks when fewer than 8 bits remain in the DRED payload. There is no arbitrary limit on the number of vectors that can be coded in a packet, but the authors do not believe that using more than a few seconds of redundancy is likely to be useful. Also, decoders MAY ignore any redundancy data beyond a certain amount.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Q0   |  dQ |X| (Ext. offset) | Offset  |Qmax | Initial state |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +
:                                                               :
+             ...               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |  Latent vectors 0,            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|  latent vector 1, ...                                         |
:                                                               :
+                                                     +-+-+-+-+-|
|                            Latent vector n-1        | unused  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Extension framing

3.1. Latent decoding

Since the DRED decoder is normative, we describe DRED from the decoder perspective, but the encoder is expected to have the corresponding behavior. DRED uses the same range coder as the rest of Opus, as described in Section 4.1 of [RFC6716]. Because the non-entropy-coded bits (Q0, dQ, ...) do not amount to an integer number of bytes, it is simpler to code them using the range coder. The result is the same for those bits, but it ensures that the complete DRED payload is an integer number of bytes (which is important to handle the end condition).

The initial state and latent vectors are handled in the same way, both coded one dimension at a time. For each dimension, the decoder uses the quantization tables to determine the _r_ and _p0_ parameters. If r=0 or p0=255 for the current symbols and quantizer, then no symbol is decoded and the decoded quantized value is 0. Otherwise, decoding proceeds as follows.

The first symbol decoded determines whether the quantized index is zero, positive, or negative (in that order). The decoder uses the pdf {2*p0_{i,q}, 256-p0_{i,q}, 256-p0_{i,q}}/512. If the value is non-zero, a second symbol is decoded. We start by generating an "inverse cdf" in Q15:

```
                / 32768                                  , if i < 0
                |
                | MAX(7, 128*r_{i,q})                    , if i = 0
     icdf(i) = <
                | MAX(7-i, (icdf[i-1]*r_{i,q})//32768) , if 0 < i < 7
                |
                \ 0                                      , i>= 7
```

where // denotes the truncating integer division. The pdf is then given by pdf[i] = icdf[i-1]-icdf[i]. If the decoded symbol equals 7, then another symbol is decoded and added to the 7 already decoded. The process is repeated until the decoded symbol is different from 7. At that point, the sign is applied and the decoded value is equal to quantized_index*256/s_{i,q}.

4. Conformance

As for the Opus specification, we wish to allow the greatest possible choice of freedom in implementing the DRED specification. For that reason, conformance is defined through the DRED decoder only. The two decoder components -- the feature decoder and vocoder -- are handled separately, and differently from each other.

4.1. DRED Feature Decoding

DRED acoustic feature decoding is strictly defined. The decoder implementation MUST use the same weights provided in this specification. The DNN weights can be further quantized and the exact implementation of the DNN arithmetic (including activation functions) can be approximated, provided that they comply with the following test. An alternate DNN model is not allowed, as it would be easy to overfit a model to the test.

4.2. Vocoder

While the vocoder that synthesizes the audio from the decoded acoustic features isn’t normative, defining how the vocoder behaves helps defining the meaning of the features themselves.

We provide a set of test vectors where the input file contains acoustic features and the corresponding original audio from which the features were computed. To verify an vocoder implementation, we will provide a tool [TBD] that compares the vocoder output to the reference output. The comparison thresholds are meant to accept any vocoder that sounds sufficiently similar. Also, no waveform-domain comparison is possible since the acoustic features do not capture phase information.

The test vector material MUST NOT be used to train the vocoder since there would be a risk of overfitting.

5. IANA Considerations

[Note: Until the IANA performs the actions described below, implementers should use 126 instead of 32 as the extension number. Moreover, the DRED payload temporarily uses a two-byte prefix for compatibility: a ’D’ character, followed by a version number (currently 10).]

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

5.1. 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 received 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.

5.2. 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.

6. 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 treatement 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 dymanically 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.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7587] Spittka, J., Vos, K., and JM. Valin, "RTP Payload Format for the Opus Speech and Audio Codec", RFC 7587, DOI 10.17487/RFC7587, June 2015, <https://www.rfc-editor.org/info/rfc7587>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC6716] Valin, JM., Vos, K., and T. Terriberry, "Definition of the Opus Audio Codec", RFC 6716, DOI 10.17487/RFC6716, September 2012, <https://www.rfc-editor.org/info/rfc6716>.

[opus-extension] Terriberry, T.B. and J.-M. Valin, "Extension Formatting for the Opus Codec (draft-ietf-mlcodec-opus-extension)", October 2023.

7.2. Informative References

[RFC6562] Perkins, C. and JM. Valin, "Guidelines for the Use of Variable Bit Rate Audio with Secure RTP", RFC 6562, DOI 10.17487/RFC6562, March 2012, <https://www.rfc-editor.org/info/rfc6562>.

[dred-paper] Valin, J.-M., Buethe, J., and A. Mustafa, "Low-Bitrate Redundancy Coding of Speech Using a Rate-Distortion-Optimized Variational Autoencoder", 2023, <https://arxiv.org/abs/2212.04453>.

Authors’ Addresses

Jean-Marc Valin
Google
Canada
Email: jeanmarcv@google.com

Jan Buethe
Xiph.Org Foundation
Germany
Email: jan.buethe@googlemail.com