Internet Engineering Task Force
Internet-Draft
Updates: 6716 (if approved)
Intended status: Standards Track
Expires: 23 July 2026
JM. Valin
Google
J. Buethe
Meta Platforms, Inc.
19 January 2026

Deep Audio Redundancy (DRED) Extension for the Opus Codec
draft-ietf-mlcodec-opus-dred-05

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

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:

```
           /
           | p0                 ,   if X = 0
           |
  P(X) = <              |X|
           | (1 - p0) * r       ,   if X != 0
           | ---------------
           \   2 * (1 - r)
```

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/cdense* denote fully-connected layers, glu1..glu5 denote gated linear units (GLUs), and cat() denote tensor concatenation. All GRU layers have 64 outputs (number of neurons) and all convolutional layers and cdense* 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 50-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 50 inputs and 128 output, and D2 has 128 inputs and 320 (5*64) 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 a 25-dimensional vector constructed from the decoded 24-dimensional latent vector for a particular 40-ms chunk, to which we append the value of Q0/8-1, where Q0 is the initial quantizer (see below). From there, the DenseNet structure can be expressed as:

```
t1 = dense1(Z)
t2 = cat(t1, conv1(cdense1(t1)))
t3 = cat(t2, glu1(gru1(t2)))
t4 = cat(t3, conv2(cdense2(t3)))
t5 = cat(t4, glu2(gru2(t4)))
t6 = cat(t5, conv3(cdense3(t5)))
t7 = cat(t6, glu3(gru3(t6)))
t8 = cat(t7, conv4(cdense4(t7)))
t9 = cat(t8, glu4(gru4(t8)))
t10 = cat(t9, conv5(cdense5(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	219	191	168	151	138	135	3	2	2	1	1	1	1	1	1
1	255	213	182	158	139	123	108	91	83	74	64	54	44	38	35	32
2	255	200	156	120	90	65	46	32	20	1	2	0	0	0	0	0
3	255	217	187	164	148	140	152	2	2	2	1	1	1	1	1	1
4	255	216	185	161	142	127	110	4	3	2	2	1	1	1	1	1
5	255	210	175	147	126	109	97	86	79	85	74	9	3	2	2	1
6	255	215	183	158	139	123	108	2	2	1	1	1	1	1	0	0
7	255	208	170	140	117	99	84	73	62	55	48	43	38	33	29	25
8	255	208	171	141	116	97	86	0	0	0	0	0	0	0	0	0
9	255	208	170	140	116	97	83	70	59	51	43	37	32	27	23	20
10	255	208	171	141	118	99	85	73	62	53	45	34	12	8	7	6
11	255	213	179	153	134	119	111	107	103	96	28	0	0	0	0	0
12	255	213	179	153	132	117	106	97	92	91	88	10	5	3	2	2
13	255	207	169	139	115	97	82	70	60	52	45	40	35	31	27	24
14	255	210	174	146	124	106	93	81	71	66	55	15	3	2	2	1
15	255	218	188	165	148	134	126	3	2	2	1	1	1	1	1	1
16	255	214	181	155	140	159	117	2	2	2	1	1	1	1	1	0
17	255	210	174	146	124	107	93	81	71	66	61	42	8	1	1	1
18	255	217	187	163	144	129	114	4	3	2	2	1	1	1	1	1

19	255	212	179	152	132	116	105	100	95	104	11	2	1	1	1	1
20	255	213	180	154	134	119	108	103	93	87	64	11	4	3	2	2
21	255	211	176	148	126	109	96	83	77	81	73	5	2	1	1	1
22	255	214	182	156	136	119	134	3	2	2	1	1	1	1	1	1
23	255	199	154	119	90	66	46	3	2	2	1	1	1	1	1	1
24	255	218	189	167	151	148	166	3	2	2	1	1	1	1	1	1

Table 2: Scale values for latent

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	37	50	63	79	101	136	255	255	255	255	255	255	255	255	255	255
1	72	98	126	157	192	226	255	255	255	255	255	255	255	255	255	255
2	2	2	2	2	3	4	7	10	13	255	255	255	255	255	255	255
3	25	36	49	68	98	173	255	255	255	255	255	255	255	255	255	255
4	24	33	42	54	69	90	120	255	255	255	255	255	255	255	255	255
5	14	17	19	23	27	32	41	49	65	150	184	255	255	255	255	255
6	25	32	40	50	63	83	117	255	255	255	255	255	255	255	255	255
7	0	0	0	1	2	3	5	8	11	16	20	26	33	41	48	57
8	10	16	23	33	47	94	255	255	255	255	255	255	255	255	255	255
9	0	1	1	1	1	1	2	2	3	3	4	5	6	7	8	10
10	0	1	3	4	6	7	9	13	14	23	46	96	224	255	255	255
11	29	33	38	44	52	64	86	123	173	255	255	0	255	255	255	255
12	7	15	23	32	42	54	71	100	148	255	255	255	255	255	255	255
13	0	0	1	2	2	4	5	7	11	15	20	26	34	44	53	63
14	11	14	17	20	23	27	33	38	46	74	92	255	255	255	255	255

15	29	40	51	65	83	111	196	255	255	255	255	255	255	255	255	255
16	29	36	46	64	112	255	255	255	255	255	255	255	255	255	255	255
17	2	6	9	13	16	21	26	30	36	65	106	105	255	255	255	255
18	22	32	43	55	70	90	116	255	255	255	255	255	255	255	255	255
19	17	22	26	32	38	45	56	90	137	255	255	255	255	255	255	255
20	4	19	37	57	81	111	157	251	255	255	255	255	255	255	255	255
21	15	17	20	23	26	30	36	44	69	160	242	255	255	255	255	255
22	23	30	38	48	61	85	255	255	255	255	255	255	255	255	255	255
23	15	16	20	27	40	63	96	255	255	255	255	255	255	255	255	255
24	30	40	53	72	104	196	255	255	255	255	255	255	255	255	255	255

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	51	38	28	20	14	10	3	0	0	0	0	0	0	0	0	0
1	5	3	2	2	1	1	1	0	0	0	0	0	0	0	0	0
2	248	246	243	239	234	226	215	198	170	0	0	0	0	0	0	0
3	50	36	25	17	12	8	0	0	0	0	0	0	0	0	0	0
4	67	51	37	26	18	11	6	0	0	0	0	0	0	0	0	0
5	110	93	76	60	46	34	26	19	14	13	6	0	0	0	0	0
6	61	44	30	19	12	7	3	0	0	0	0	0	0	0	0	0
7	193	183	171	158	144	130	117	103	89	77	65	55	44	33	23	16
8	73	52	33	17	6	1	0	0	0	0	0	0	0	0	0	0
9	228	222	216	208	199	190	180	168	155	143	129	116	101	84	70	56
10	168	153	136	119	101	85	69	55	41	30	20	8	0	0	0	0

11	87	70	54	41	30	23	18	14	10	4	0	0	0	0	0	0
12	92	77	64	53	45	37	28	19	11	5	0	0	0	0	0	0
13	197	187	175	162	149	136	123	111	99	89	80	72	63	51	40	31
14	125	105	86	67	51	37	27	18	12	10	5	0	0	0	0	0
15	51	37	25	16	10	6	3	0	0	0	0	0	0	0	0	0
16	35	23	15	10	8	0	0	0	0	0	0	0	0	0	0	0
17	131	114	96	79	64	50	39	28	20	16	11	1	0	0	0	0
18	67	52	38	27	19	12	6	0	0	0	0	0	0	0	0	0
19	95	76	58	43	31	21	15	12	8	3	0	0	0	0	0	0
20	79	65	49	35	24	16	10	5	3	1	0	0	0	0	0	0
21	118	100	82	65	51	38	28	19	14	12	4	0	0	0	0	0
22	55	39	25	16	9	5	1	0	0	0	0	0	0	0	0	0
23	156	138	119	99	77	49	23	0	0	0	0	0	0	0	0	0
24	43	31	23	16	13	7	0	0	0	0	0	0	0	0	0	0

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	171	190	207	221	233	243	253	255	255	255	255	255	255	255	255	255
1	251	253	254	254	255	255	255	255	255	255	255	255	255	255	255	255
2	4	5	6	8	11	15	21	31	48	255	255	255	255	255	255	255
3	158	178	198	216	232	246	255	255	255	255	255	255	255	255	255	255
4	146	165	183	200	215	230	243	255	255	255	255	255	255	255	255	255
5	115	130	145	161	176	190	203	215	226	239	248	255	255	255	255	255
6	140	159	178	197	214	230	245	255	255	255	255	255	255	255	255	255

7	63	73	83	92	102	111	121	132	143	155	166	176	189	202	214	224
8	120	141	164	189	216	242	255	255	255	255	255	255	255	255	255	255
9	14	17	21	26	30	36	42	49	57	65	74	84	95	109	123	136
10	66	75	85	97	109	122	135	150	164	183	209	242	255	255	255	255
11	141	157	173	188	202	215	226	236	244	252	255	255	255	255	255	255
12	142	160	177	193	207	219	228	237	245	251	255	255	255	255	255	255
13	51	59	69	79	90	101	113	126	140	153	167	179	192	205	216	225
14	84	98	112	128	144	161	177	194	210	227	242	255	255	255	255	255
15	158	177	195	211	226	238	250	255	255	255	255	255	255	255	255	255
16	174	194	213	230	246	255	255	255	255	255	255	255	255	255	255	255
17	100	113	127	141	155	169	182	195	208	224	237	251	255	255	255	255
18	152	170	187	203	217	230	241	255	255	255	255	255	255	255	255	255
19	117	132	149	165	181	196	210	225	239	252	255	255	255	255	255	255
20	172	191	207	221	232	240	246	251	253	255	255	255	255	255	255	255
21	110	123	137	152	166	181	194	208	223	239	250	255	255	255	255	255
22	138	158	179	199	218	235	255	255	255	255	255	255	255	255	255	255
23	93	109	129	153	179	207	233	255	255	255	255	255	255	255	255	255
24	169	189	208	224	238	249	255	255	255	255	255	255	255	255	255	255

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	208	171	141	118	100	86	74	63	54	48	42	35	30	26	23
1	99	88	79	71	64	58	54	54	58	255	220	175	141	113	95	92
2	255	210	174	146	123	106	91	80	69	61	57	46	41	41	41	38

3	255	208	171	142	119	100	86	73	62	53	45	39	38	36	33	31
4	108	96	86	79	73	69	68	72	79	114	152	227	255	225	194	168
5	255	206	167	136	112	92	77	65	54	47	40	35	31	26	23	20
6	255	206	167	137	112	93	78	66	55	47	41	35	30	26	23	20
7	255	209	173	145	123	105	92	81	72	67	67	64	58	54	49	45
8	255	212	179	153	132	115	104	80	75	68	52	34	26	23	20	18
9	255	210	174	146	124	107	95	78	69	61	50	43	39	37	35	34
10	255	210	174	146	125	107	94	82	80	78	69	60	53	47	43	39
11	255	207	169	139	116	97	83	71	62	55	49	44	39	35	32	30
12	209	190	173	158	143	131	122	128	128	180	209	249	255	228	202	181
13	255	207	170	140	117	98	83	71	62	54	50	47	44	42	39	37
14	136	123	111	101	92	87	93	167	158	202	221	255	233	198	171	150
15	255	203	159	124	95	72	54	41	37	56	44	34	26	20	15	12
16	255	212	177	150	129	113	101	88	77	65	49	40	34	29	25	23
17	255	218	190	167	150	139	138	157	139	122	107	95	83	73	66	63
18	44	40	37	34	31	29	27	28	27	255	204	161	127	99	77	61
19	255	206	168	137	113	95	80	68	57	50	44	39	34	29	26	23
20	147	134	122	110	101	94	92	125	118	151	183	252	255	224	195	172
21	169	154	140	128	118	110	106	105	110	147	173	225	255	232	205	179
22	255	198	154	119	91	69	52	39	32	35	27	21	16	13	10	8
23	255	209	172	143	121	103	90	79	70	63	58	53	49	45	43	43
24	255	205	167	136	112	93	78	66	56	49	43	38	33	30	26	24
25	241	219	199	181	165	151	141	134	127	182	210	255	234	202	180	170
26	255	211	176	149	127	110	97	81	72	60	49	40	30	26	23	20

27	255	208	171	142	119	100	85	73	63	56	50	45	40	35	33	32
28	255	207	169	139	116	98	84	71	60	52	45	40	34	28	24	21
29	255	212	178	151	131	115	103	86	76	66	55	41	29	23	19	17
30	255	218	188	165	147	135	133	157	138	119	104	91	79	68	61	56
31	255	210	174	145	123	105	93	84	77	76	68	60	54	50	46	42
32	255	207	169	138	114	95	81	69	59	52	46	40	34	30	26	23
33	255	208	172	144	121	104	90	77	64	42	37	27	22	19	17	15
34	255	227	205	186	172	161	160	155	140	117	106	95	74	72	69	62
35	255	207	169	138	115	96	81	68	57	49	42	36	30	25	22	18
36	255	209	173	144	122	103	90	77	65	55	51	46	38	33	29	26
37	255	212	179	153	133	118	111	111	101	102	99	90	76	65	57	50
38	255	217	186	162	143	130	129	147	132	120	106	94	81	72	71	74
39	255	208	169	139	115	96	80	65	51	40	34	30	26	23	21	18
40	72	66	62	57	52	47	45	47	52	255	222	177	144	116	96	84
41	255	207	170	140	117	98	84	72	63	55	48	42	37	32	30	29
42	255	211	177	150	130	114	105	94	78	54	35	16	12	11	9	8
43	255	206	168	137	113	94	79	66	56	49	42	37	31	26	22	19
44	255	205	165	135	111	91	76	64	54	46	39	33	29	24	21	18
45	61	56	51	47	43	40	37	36	39	255	212	166	133	106	84	68
46	255	210	174	147	125	108	96	87	85	81	73	64	57	51	45	41
47	255	207	169	139	116	99	86	73	63	57	49	42	37	32	29	27
48	255	205	166	136	112	92	77	65	55	47	40	34	29	25	22	19
49	255	212	173	140	115	95	80	68	57	49	41	35	30	25	21	18

Table 6: Scale values for state

k	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0	32	28	24	22	20	20	21	15	17	12	12	16	15	18	24	34
1	255	255	255	255	255	255	255	255	255	1	19	16	26	38	69	245
2	11	13	15	17	20	23	26	31	36	38	51	54	85	196	255	255
3	18	17	18	19	20	23	27	26	31	33	38	64	187	255	255	255
4	255	255	255	255	255	255	255	255	255	75	16	0	0	16	20	21
5	21	17	12	9	6	4	3	3	0	0	4	8	14	17	19	21
6	14	10	7	4	3	2	2	3	1	4	5	7	9	12	13	15
7	0	2	8	13	20	28	38	49	71	129	255	255	255	255	255	255
8	6	12	19	27	35	45	58	34	66	126	255	255	255	255	255	255
9	0	0	3	15	30	50	79	95	130	181	207	249	255	255	255	255
10	0	0	6	13	21	30	45	69	151	255	255	255	255	255	255	255
11	5	8	10	13	15	18	21	26	34	40	49	63	77	92	112	145
12	255	255	255	255	255	255	255	255	255	102	45	2	18	31	43	58
13	3	6	9	12	16	19	20	29	41	58	92	141	199	255	255	255
14	255	255	255	255	255	255	255	160	46	15	0	2	19	23	28	35
15	10	12	13	12	9	6	0	0	3	0	10	6	2	3	3	3
16	3	7	9	12	13	14	11	0	16	26	27	91	255	255	255	255
17	33	34	31	24	15	6	0	0	16	19	23	29	35	49	73	137
18	255	255	255	255	255	255	255	255	255	0	0	0	2	6	9	12
19	26	21	17	14	12	12	14	15	12	17	20	26	32	37	44	52
20	255	255	255	255	255	255	255	255	255	44	11	10	2	16	24	32
21	255	255	255	255	255	255	255	255	255	255	91	13	1	18	26	30

22	15	13	11	9	7	5	3	5	8	0	4	11	19	33	56	102
23	4	7	10	13	16	20	26	34	42	50	55	76	101	137	204	255
24	6	5	5	6	6	8	10	13	18	25	31	40	49	56	63	72
25	255	255	255	255	255	255	255	255	255	24	12	3	20	31	49	101
26	8	11	14	18	22	27	33	22	39	61	98	255	255	255	255	255
27	7	9	11	13	15	17	19	21	23	35	46	51	64	82	113	178
28	20	18	16	15	15	17	21	13	15	17	20	18	15	16	19	23
29	0	2	7	11	15	19	21	21	38	63	138	255	255	255	255	255
30	18	19	17	11	4	0	0	7	19	20	23	27	33	43	59	92
31	0	2	4	7	10	15	24	38	58	97	117	145	193	255	255	255
32	9	9	9	9	9	10	12	17	18	24	28	26	32	39	47	59
33	18	17	17	18	19	21	25	25	29	27	255	255	255	255	255	255
34	32	39	39	32	19	1	0	6	20	20	33	46	49	197	255	255
35	8	9	8	7	6	4	4	2	7	6	4	5	6	7	8	9
36	8	10	12	14	17	20	23	22	31	51	125	255	255	255	255	255
37	37	34	30	25	20	14	13	32	19	6	9	16	15	20	28	42
38	72	65	55	41	24	3	0	0	12	14	18	25	31	54	142	255
39	0	0	1	6	11	15	19	26	44	255	255	255	255	255	255	255
40	255	255	255	255	255	255	255	255	255	17	37	23	29	34	50	92
41	9	11	12	13	14	16	19	23	27	32	34	42	51	64	88	139
42	0	1	9	15	22	28	32	45	51	57	255	255	255	255	255	255
43	33	28	22	18	14	10	9	5	9	8	11	11	11	11	11	12
44	5	4	3	2	2	3	3	3	1	2	3	3	2	2	2	2
45	255	255	255	255	255	255	255	255	255	0	15	8	14	19	26	38

46	0	0	3	11	18	27	40	73	166	255	255	255	255	255	255	255
47	12	13	13	12	10	5	0	7	14	11	13	16	22	34	53	95
48	15	11	7	4	2	1	1	3	0	1	0	0	2	5	7	10
49	12	14	14	13	11	9	8	9	12	8	3	4	3	2	3	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	197	186	174	161	147	134	121	106	92	81	66	51	36	24	14
1	0	0	0	0	0	0	0	0	0	85	64	45	25	11	3	0
2	141	124	107	90	74	60	48	37	26	19	15	7	3	1	0	0
3	141	123	104	85	67	51	37	25	15	8	4	1	0	0	0	0
4	2	1	0	0	0	0	0	0	0	8	42	84	95	77	62	48
5	226	219	211	202	191	180	168	155	141	128	114	100	85	69	55	43
6	226	220	212	204	195	184	173	160	147	134	120	106	91	77	63	50
7	98	79	60	44	32	22	16	11	8	5	1	0	0	0	0	0
8	103	83	65	49	38	27	20	7	3	1	0	0	0	0	0	0
9	78	68	61	49	35	24	16	10	6	3	2	1	0	0	0	0
10	79	61	44	29	18	11	7	3	2	0	0	0	0	0	0	0
11	160	146	132	117	103	90	79	67	57	48	39	30	22	16	11	7
12	1	0	0	0	0	0	0	0	0	2	12	36	32	20	11	5
13	138	121	103	86	71	58	48	41	37	28	17	10	6	4	2	1
14	0	0	0	0	0	0	0	6	15	42	55	67	52	37	25	15
15	251	250	248	246	243	240	235	228	225	235	230	223	214	202	187	169
16	114	97	79	63	49	35	27	23	12	5	1	0	0	0	0	0

17	120	104	90	79	72	68	70	82	65	52	39	28	18	9	3	1
18	4	2	1	0	0	0	0	0	2	192	179	163	143	121	97	72
19	203	193	181	167	153	138	123	107	91	77	63	51	38	26	17	11
20	0	0	0	0	0	0	0	0	0	9	28	53	57	40	28	17
21	1	0	0	0	0	0	0	0	0	0	7	43	59	44	31	21
22	232	225	217	207	195	179	159	137	120	128	107	84	61	40	24	12
23	134	117	101	86	72	59	48	39	32	26	20	16	11	7	3	1
24	192	181	169	156	143	130	117	104	92	80	69	58	48	39	31	25
25	1	0	0	0	0	0	0	0	0	7	16	30	19	10	4	1
26	111	93	76	61	47	36	28	14	11	5	2	0	0	0	0	0
27	175	161	145	129	113	97	82	67	54	43	34	25	17	10	6	2
28	223	216	208	199	189	178	167	155	142	128	115	103	86	69	54	40
29	104	86	69	54	41	31	23	11	6	3	0	0	0	0	0	0
30	136	121	107	96	87	81	79	93	76	62	49	36	24	13	6	2
31	102	86	72	60	50	41	35	29	22	14	1	0	0	0	0	0
32	201	190	178	164	149	133	118	101	85	70	56	45	31	20	11	6
33	119	100	81	63	48	34	23	15	7	1	0	0	0	0	0	0
34	88	75	64	56	54	55	54	50	36	24	15	8	2	0	0	0
35	247	244	242	239	235	232	227	223	216	210	203	195	186	175	163	151
36	114	95	76	58	42	29	19	12	5	1	0	0	0	0	0	0
37	144	128	112	97	85	76	70	63	59	64	61	49	37	24	14	7
38	94	76	63	53	48	49	48	60	46	37	27	18	9	3	0	0
39	130	111	92	72	52	36	22	10	2	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	99	79	61	40	23	10	2

41	181	168	153	137	121	104	88	72	57	44	32	21	12	6	2	1
42	103	83	64	48	35	24	19	10	4	0	0	0	0	0	0	0
43	239	236	231	226	220	214	207	198	189	181	170	160	147	131	115	100
44	246	244	241	238	234	229	224	219	212	206	198	189	180	169	158	146
45	0	0	0	0	0	0	0	0	0	142	122	101	77	54	33	18
46	73	56	40	26	15	8	6	4	2	0	0	0	0	0	0	0
47	187	173	158	143	127	113	101	83	66	58	44	32	21	11	5	1
48	232	226	220	212	204	194	184	173	161	149	136	122	108	93	79	65
49	254	254	253	253	252	251	251	250	248	247	246	244	242	239	236	233

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	27	32	38	45	53	62	71	80	91	103	112	126	142	160	178	197
1	255	255	255	255	255	255	255	255	255	109	128	149	175	203	230	253
2	68	81	96	112	128	144	160	176	192	203	214	230	244	253	255	255
3	67	79	94	109	126	143	160	177	195	210	225	242	254	255	255	255
4	253	255	255	255	255	255	255	255	255	210	152	109	100	115	130	145
5	19	23	26	30	35	41	49	57	66	75	85	96	109	123	137	151
6	29	35	40	46	52	59	65	73	81	89	98	109	119	130	141	152
7	100	116	134	154	174	193	210	225	238	249	254	255	255	255	255	255
8	94	110	128	147	168	187	202	213	230	247	255	255	255	255	255	255
9	152	169	188	206	221	232	240	246	250	253	254	255	255	255	255	255
10	115	131	151	170	190	209	225	241	252	255	255	255	255	255	255	255
11	82	96	111	127	142	157	171	185	198	207	217	226	233	240	245	249

12	255	255	255	255	255	255	255	255	255	235	201	160	166	185	204	221
13	79	94	110	127	145	163	179	197	214	228	238	246	250	252	254	255
14	255	255	255	255	255	255	255	230	194	153	138	125	141	159	176	194
15	2	3	4	5	6	8	11	14	16	11	13	17	22	29	37	48
16	98	112	126	141	154	165	173	180	200	219	237	254	255	255	255	255
17	81	93	104	114	120	124	122	111	127	141	156	171	189	209	227	244
18	251	254	255	255	255	255	255	255	253	34	42	52	65	80	99	120
19	53	62	68	76	84	93	104	114	124	136	147	160	174	189	203	215
20	255	255	255	255	255	255	255	255	255	208	171	140	135	154	172	189
21	255	255	255	255	255	255	255	255	255	254	214	151	134	150	167	183
22	16	20	25	32	40	50	64	82	97	86	105	129	157	186	214	237
23	99	114	129	145	160	174	188	201	212	220	227	236	243	249	253	255
24	64	75	87	100	113	126	139	152	164	176	187	198	208	217	225	231
25	255	255	255	255	255	255	255	255	255	212	193	169	187	207	226	243
26	88	103	121	139	158	174	189	198	217	236	250	255	255	255	255	255
27	47	57	69	81	94	108	123	138	153	170	185	198	213	227	238	248
28	17	21	25	30	36	42	49	57	66	75	85	94	107	123	139	155
29	106	120	136	151	166	179	189	204	221	238	253	255	255	255	255	255
30	70	80	90	99	107	112	113	102	116	130	144	160	178	198	217	234
31	122	138	155	172	187	202	214	225	233	240	245	250	253	255	255	255
32	29	35	43	51	61	71	82	95	108	122	136	149	167	185	202	218
33	81	96	112	129	146	163	179	194	215	242	255	255	255	255	255	255
34	107	120	129	136	139	138	138	143	160	178	195	212	233	254	255	255
35	5	6	7	9	11	13	15	17	21	24	28	32	38	44	52	59

36	85	100	117	134	153	170	186	200	219	238	252	255	255	255	255	255
37	64	75	87	98	109	117	122	129	133	128	132	143	159	178	196	215
38	106	117	130	139	145	144	145	132	148	158	173	189	207	228	248	255
39	74	87	103	121	140	160	180	206	235	255	255	255	255	255	255	255
40	255	255	255	255	255	255	255	255	255	97	114	131	154	180	206	232
41	58	68	79	90	101	113	125	138	151	165	177	192	209	223	236	246
42	99	111	129	147	164	180	189	205	223	246	255	255	255	255	255	255
43	8	10	13	15	19	22	26	31	36	41	47	54	62	73	84	96
44	5	6	8	9	11	14	16	19	23	27	31	36	42	48	55	62
45	255	255	255	255	255	255	255	255	255	66	79	95	115	139	165	193
46	120	137	155	175	195	212	228	243	253	255	255	255	255	255	255	255
47	37	45	55	65	76	86	95	110	126	134	150	165	182	202	222	240
48	13	16	19	23	28	33	39	45	53	61	69	79	90	102	114	127
49	1	1	1	2	2	2	3	3	4	4	5	6	7	9	10	12

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., Mustafa, A., and M. Klingbeil, "DRED: Deep REDundancy Coding of Speech Using a Rate-Distortion-Optimized Variational Autoencoder", IEEE Journal of Selected Topics in Signal Processing vol. 18, no. 8, DOI 10.1109/JSTSP.2024.3482972, 2024, <https://arxiv.org/abs/2212.04453>.

Authors’ Addresses

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

Jan Buethe
Meta Platforms, Inc.
Germany
Email: jan.buethe@googlemail.com