Audio/Video Transport Working Group S. Ikonin
Internet Draft SPIRIT DSP
Intended status: Informational February 02, 2010

RTP Payload Format for IP-MR Speech Codec draft-ietf-avt-rtp-ipmr-11.txt

Status of this Memo

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

This Internet-Draft will expire on June 02, 2010.

Abstract

This document specifies the payload format for packetization of SPIRIT IP-MR encoded speech signals into the Real-time Transport Protocol (RTP). The payload format supports transmission of multiple frames per payload and introduced redundancy for robustness against packet loss.

Table of Contents

1. Introduction....3
2. IP-MR Codec Description....3
3. Payload Format....4
3.1. RTP Header Usage....4
3.2. Payload Format Structure....5
3.3. Payload Header....5
3.4. Speech Table of Contents....6
3.5. Speech Data....7
3.6. Redundancy Header....7
3.7. Redundancy Table of Contents....8
3.8. Redundancy Data....9
4. Payload Examples....9
4.1. Payload Carrying a Single Frame....9
4.2. Payload Carrying Multiple Frames with Redundancy....10
5. Media Type Registration....11
5.1. Registration of media subtype audio/ip-mr_v2.5....11
5.2. Mapping Media Type Parameters into SDP....12
6. Security Considerations....13
7. Congestion Control....13
8. IANA Considerations....14
9. Normative References....14
10. Author(s) Information....15
11. Disclaimer....15
12. Legal Terms....15
APPENDIX A. RETRIEVING FRAME INFORMATION....17
A.1. get_frame_info.c....17
Authors' Addresses....19

1. Introduction

This document specifies the payload format for packetization of SPIRIT IP-MR encoded speech signals into the Real-time Transport Protocol (RTP). The payload format supports transmission of multiple frames per payload and introduced redundancy for robustness against packet loss.

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

2. IP-MR Codec Description

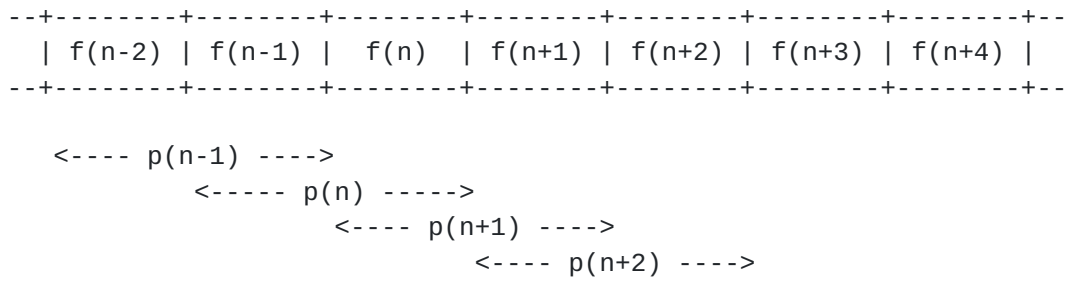
The IP-MR codec is scalable adaptive multi-rate wideband speech codec designed by SPIRIT for use in IP based networks. These codec is suitable for real time communications such as telephony and videoconferencing.

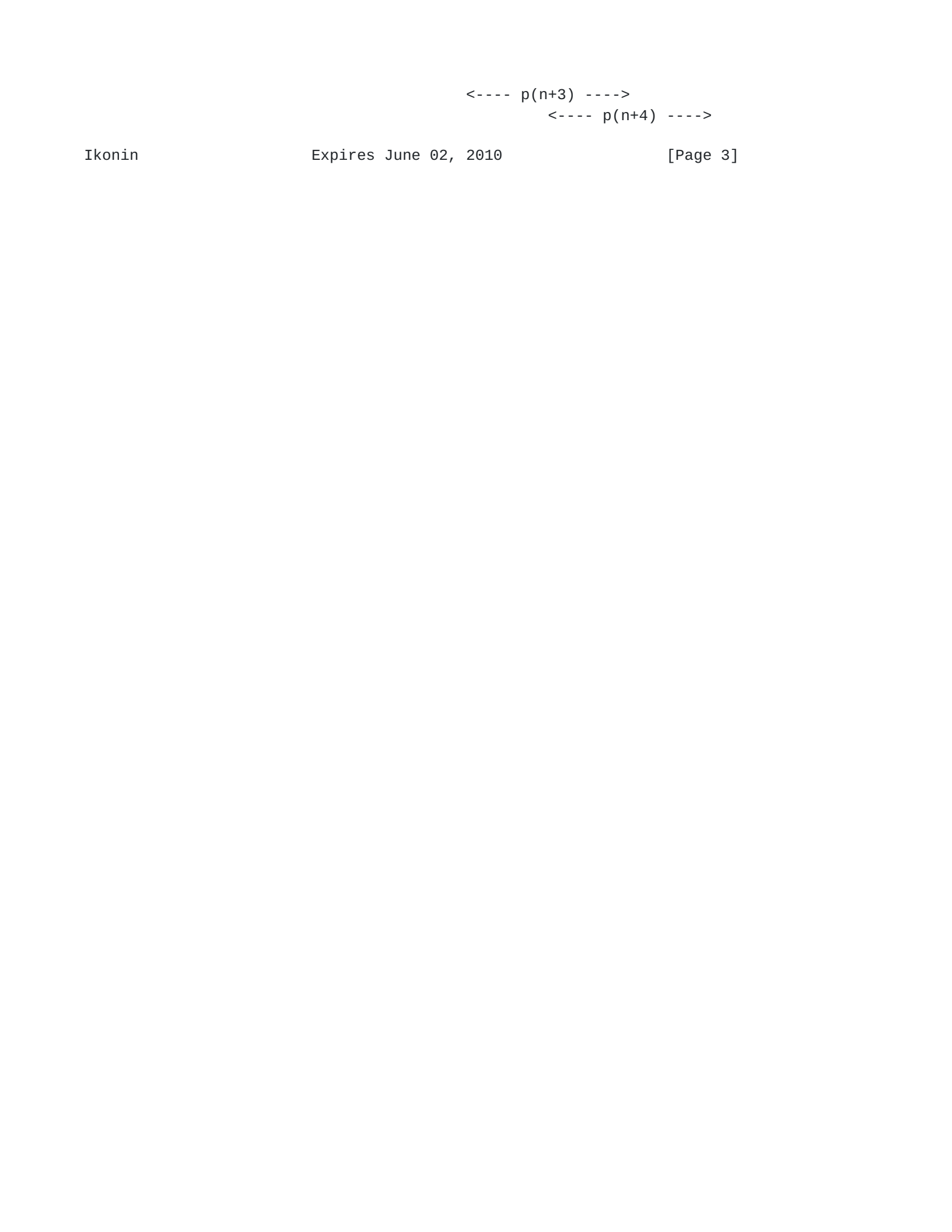
The codec operates on 20 ms frames at 16 kHz sampling rate and has an algorithmic delay of 25ms.

The IP-MR supports six wide band speech coding modes with respective bit rates ranging from about 7.7 to about 34.2 kbps. The coding mode can be changed at any 20 ms frame boundary making possible to dynamically adjust the speech encoding rate during a session to adapt to the varying transmission conditions.

The coded frame consists of multiple coding layers - base (or core) layer and several enhancement layers which are coded independently. Only the core layer is mandatory to decode understandable speech and upper layers provide quality enhancement. These enhancement layers may be omitted and remaining base layer can be meaningfully decoded without artifacts. This makes the bit stream scalable and allows to reduce bit rate during transmission without re-encoding.

This memo specifies an optional form of redundancy coding within RTP for protection against packet loss. It is based on commonly known scheme when previously transmitted frames are aggregated together with new ones. Each frame is retransmitted once in the following RTP payload packet. f(n-2)...f(n+4) denotes a sequence of speech frames, and p(n-1)...p(n+4) is a sequence of payload packets:




<---- p(n+3) ---->
<---- p(n+4) ---->

But because of the scalable nature of IP-MR codec there is no need to duplicate the whole previous frame - only the core layer may be retransmitted. This reduces redundancy overhead while keeping efficiency. Moreover, the speech bits encoded in core layer are divided on six classes (from A to F) of perceptual sensitivity to errors. Using these classes as introduced redundancy make possible to adjust trade-off between overhead and robustness against packet loss.

The mechanism described does not really require signaling at the session setup. The sender is responsible for selecting an appropriate amount of redundancy based on feedback about the channel conditions.

The main codec characteristics can be summarized as follows:

- Wideband, 16 kHz, speech codec
- Adaptive multi rate with six modes from about 7.7 to 34.2 kbps
- Bit rate scalable
- Variable bit rate changing in accordance with actual speech content
- Discontinuous Transmission (DTX), silence suppression and comfort noise generation
- In-band redundancy scheme for protection against packet loss

3. Payload Format

The main purpose of the payload design for IP-MR is to maximize the potential of the codec with as minimal overhead as possible. The payload format allows changing parameters of the codec (such as bit rate, level of scalability, DTX and redundancy mode) without re-negotiation at any packet boundary. This make possible dynamically adjust streaming parameters in accordance to changing network conditions. The payload format also supports aggregation of multiple consecutive frames (up to 4) in a payload. That allows controlling trade-off between delay and header overhead.

3.1. RTP Header Usage

The RTP timestamp corresponds to the sampling instant of the first sample encoded for the first frame-block in the packet. The timestamp clock frequency SHALL be 16 kHz. The duration of one frame is 20 ms, corresponding to 320 samples at 16 kHz. Thus the timestamp is increased by 320 for each consecutive frame. The timestamp is also used to recover the correct decoding order of the frame-blocks.

The RTP header marker bit (M) SHALL be set to 1 whenever the first frame-block carried in the packet is the first frame-block in a talkspurt (see definition of the talkspurt in Section 4.1 [RFC 3551]). For all other packets, the marker bit SHALL be set to zero (M=0).

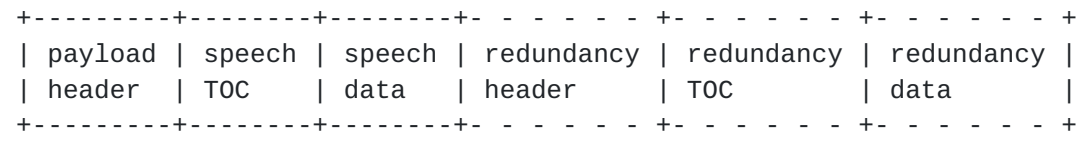
The assignment of an RTP payload type for the format defined in this memo is outside the scope of this document. The RTP profiles in use currently mandate binding the payload type dynamically for this payload format. This is basically necessary because the payload type expresses the configuration of the payload itself, i.e. basic or interleaved mode, and the number of channels carried.

The remaining RTP header fields are used as specified in [RFC 3550].

3.2. Payload Format Structure

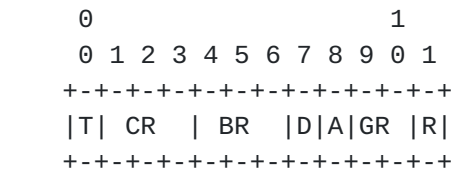
The IP-MR payload format consists of a payload header with general information about packet, a speech table of contents (TOC), and speech data. An optional redundancy section follows after speech data. The redundancy section consists of redundancy header, redundancy TOC and redundancy data payload.

The following diagram shows the standard payload format layout:



3.3. Payload Header

The payload header has the following format:



- o T (1 bit): Reserved compatibility with future extensions. MUST be set to 0.

- o CR (3 bits): coding rate of frame(s) in this packet, as per the following table:

CR	avg. bitrate
0	7.7 kbps
1	9.8 kbps
2	14.3 kbps
3	20.8 kbps
4	27.9 kbps
5	34.2 kbps
6	(reserved)
7	NO_DATA

The CR value 7 (NO_DATA) indicates that there is no speech data (and speech TOC accordingly) in the payload. This MAY be used to transmit redundancy data only. The value 6 is reserved. If receiving this value the packet MUST be discarded.

- BR (3 bits): base rate for core layer of frame(s) in this packet using the table for CR. The base rate is the lowest rate for scalability, so speech payload can be scaled down not lower than BR value. Packets with BR = 6 or BR > CR MUST be discarded.

- D (1 bit): reserved. Must be always set to 1.
  Previously, this bit indicated DTX mode availability, but in fact payload dublicates this information.

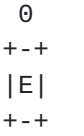
- A (1 bit): reserved. Must be always set to 1.
  Previously, this bit indicated aligned mode, but this mode has never been used and was always set to 1.

- GR (2 bits): number of frames in packet (grouping size). Actual grouping size is GR + 1, thus maximum grouping supported is 4.

- R (1 bit): redundancy presence bit. If R=1 then the packet contains redundancy information for lost packets recovery.
  In this case after speech data the redundancy section is present.

3.4. Speech Table of Contents

The speech TOC contains entries for each frame in packet (grouping size in total). Each entry contains a single field:



- E (1 bit): frame existence indicator. If set to 0, this indicates the corresponding frame is absent and the receiver should set special LOST_FRAME flag for decoder. This can be followed by the lost frame itself or by empty frames generated by the encoder during silence intervals in DTX mode.

Note that if CR flag from payload header is 7 (NO_DATA) then speech TOC is empty.

3.5. Speech Data

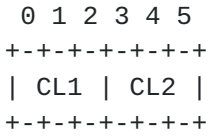
Speech data of a payload contains one or more speech frames or comfort noise frames, as specified in the speech TOC of the payload.

Each speech frame represents 20 ms of speech encoded with the rate indicated in the CR and base rate indicated in BR field of the payload header.

The size of coded speech frame is variable due to the nature of codec. The Encoder's algorithm decides what size of each frame is and returns it after encoding. In order to save bandwidth the size is not placed into payload obviously. The frame size can be determined by frame's content using a special service function specified in Appendix A. This function provides complete information about coded frame including size, number of layers, size of each layer and size of perceptual sensitive classes.

3.6. Redundancy Header

If a packet contains redundancy (R field of payload header is 1) the speech data is followed by redundancy header:



Redundancy header consists of two fields. Each field contains class specifier for amount of redundancy partly taken from the preceding packet (CL1) and pre-preceding packet (CL2), e.g. distant from the current packet by 1 and 2 packets accordingly. The values are listed in the table below:

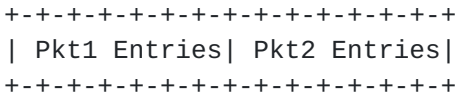
CL	amount redundancy
0	NONE
1	CLASS A
2	CLASS B
3	CLASS C
4	CLASS D
5	CLASS E
6	CLASS F
7	(reserved)

Each specifier takes 3 bits, thus the total redundancy header size is 6 bits.

These classes indicate subjective importance of bits from core layer. Class A contains the bits most sensitive to errors and lost of these bits results in a corrupted speech frame which should not be decoded without applying packet loss concealment (PLC) procedure. Class B is less sensitive than class A and so on to F. Sum of all bit classes from A to F composes core layer.

Putting some part (classes of bits) from previous frame into current packet makes possible to partially decode previous frame in case of it's lost. Than more information is delivered than less speech quality degradation will be. Flags CL1 and CL2 specify how many classes from previous frames current packet contain. E.g. CL1=3 (class C), it means that packet contains bits from classes A, B and C of previous frame. If CL1=6 (class F) then whole core layer is included.

3.7. Redundancy Table of Contents



The redundancy TOC contains entries for redundancy frames from preceding and pre-preceding packets. Each entry takes 1 bit like speech TOC entry (3.3):

o E (1 bit): frame existence indicator. If set to 0, this indicates

the corresponding frame is absent.

- For each preceding and pre-preceding packet the number of entries is equal to the grouping size of the current packet. E.g. maximum number of entries is 4*2 = 8.

- If class specifier in the redundancy header is CL=0 (NO_DATA) then there is no entries for corresponding packet redundancy.

3.8. Redundancy Data

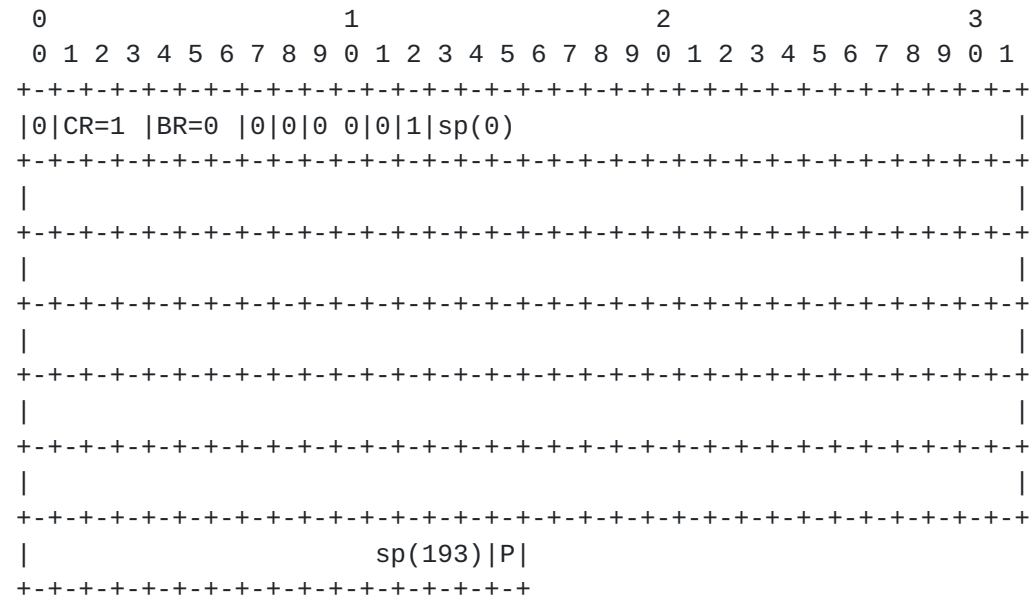
Redundancy data of a payload contains redundancy information for one or more speech frames or comfort noise frames that may be lost during transition, as specified in the redundancy TOC of the payload. Actually redundancy is the most important part of preceding frames representing **20 ms of speech. This data MAY be used for partial reconstruction of** lost frames. The amount of available redundancy is specified by CL flag in redundancy header section (3.5). This flag SHOULD be passed to decoder. The size of redundancy frame is variable and can be obtained using service function specified in Appendix A.

4. Payload Examples

A few examples to highlight the payload format follow.

4.1. Payload Carrying a Single Frame

The following diagram shows a standard IP-MR payload carrying a single speech frame without redundancy:



In the payload the speech frame is not damaged at the IP origin (E=1), the coding rate is 9.7 kbps(CR=1), the base rate is 7.8 kbps (BR=0), and the DTX mode is off. There is no byte alignment (A=0) and no redundancy (R=0). The encoded speech bits - s(0) to s(193) - are placed immediately after TOC. Finally, one zero bit is added at the end as padding to make the payload byte aligned.

4.2. Payload Carrying Multiple Frames with Redundancy

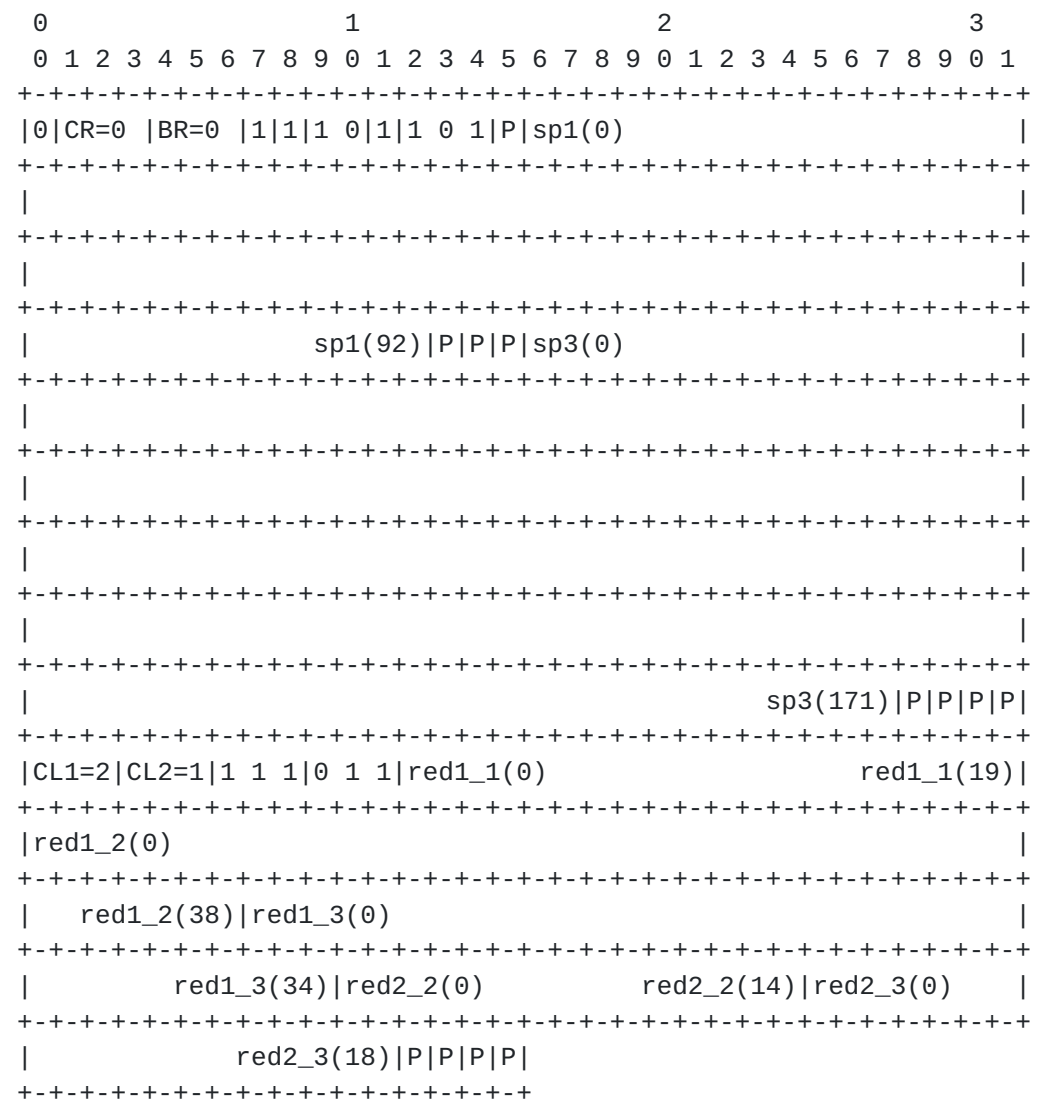
The following diagram shows a payload that contains three frames, one of them with no speech data. The coding rate is 7.7 kbps (CR=0), the base rate is 7.7 kbps (BR=0), and the DTX mode is on. The speech frames are byte aligned (A=1), so 1 zero bit is added at the end of the header. Besides the speech frames the payload contains six redundancy frames (three per each delayed packet).

The first speech frame consists of bits sp1(0) to sp1(92). After that 3 bits are added for byte alignment. The second frame does not contain any speech information that is represented in the payload by its TOC entry. The third frame consists of bits sp3(0) to sp3(171).

The redundancy header follows after speech data. The one-packet-delayed redundancy contains class A+B bits (CL1=2), and two-packet-delayed redundancy contains class A bits (Cl2=1). The one-packet-delayed redundancy contains three frames with 20, 39 and 35 bits respectively.

The first frame of two-packet-delayed redundancy is absent, it is represented in its TOC entry, and two other frames have sizes 15 and 19 bits.

Note that all speech frames are padded with zero bits for byte alignment.



5. Media Type Registration

This section describes the media types and names associated with this payload format.

5.1. Registration of media subtype audio/ip-mr_v2.5

Type name: audio

Subtype name: ip-mr_v2.5

Required parameters: none

Optional parameters:
* ptime: Gives the length of time in milliseconds represented by the

media in a packet. Allowed values are: 20, 40, 60 and 80.

Encoding considerations: This media type is framed binary data (see RFC 4288, Section 4.8).

Security considerations: See RFC 3550 [RFC 3550]

Interoperability considerations: none

Published specification: RFC XXXX

Applications that use this media type: Real-time audio applications like voice over IP and teleconference, and multi-media streaming.

Additional information: none

Person & email address to contact for further information:
Yury Morzeev
morzeev@spiritdsp.com

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC 3550].

Authors:
Sergey Ikonin <info@spiritdsp.com>

Change controller: IETF Audio/Video Transport working group delegated from the IESG.

5.2. Mapping Media Type Parameters into SDP

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [RFC 4566], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the IP-MR codec, the mapping is as follows:

- The media type ("audio") goes in SDP "m=" as the media name.
- The media subtype (payload format name) goes in SDP "a=rtpmap" as the encoding name. The RTP clock rate in "a=rtpmap" MUST 16000.
- The parameter "ptime" goes in the SDP "a=ptime" attributes.

Any remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the media type parameter string as a semicolon-separated list of parameter=value pairs.

Note that the payload format (encoding) names are commonly shown in upper case. Media subtypes are commonly shown in lower case. These names are case-insensitive in both places.

6. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC 3550] and in any applicable RTP profile. The main security considerations for the RTP packet carrying the RTP payload format defined within this memo are confidentiality, integrity, and source authenticity. Confidentiality is achieved by encryption of the RTP payload. Integrity of the RTP packets is achieved through a suitable cryptographic integrity protection mechanism. Such a cryptographic system may also allow the authentication of the source of the payload.

A suitable security mechanism for this RTP payload format should provide confidentiality, integrity protection, and at least source authentication capable of determining if an RTP packet is from a member of the RTP session.

Note that the appropriate mechanism to provide security to RTP and payloads following this memo may vary. It is dependent on the application, the transport, and the signaling protocol employed. Therefore, a single mechanism is not sufficient, although if suitable, usage of the Secure Real-time Transport Protocol (SRTP) [RFC 3711] is recommended. Other mechanisms that may be used are IPsec [RFC 4301] and Transport Layer Security (TLS) [RFC 5246] (RTP over TCP); other alternatives may exist.

This payload format does not exhibit any significant non-uniformity in the receiver side computational complexity for packet processing, and thus is unlikely to pose a denial-of-service threat due to the receipt of pathological data.

7. Congestion Control

The general congestion control considerations for transporting RTP data apply; see RTP [RFC 3550] and any applicable RTP profile like AVP [RFC 3551]. However, the multi-rate capability of IP-MR speech coding provides a mechanism that may help to control congestion, since the bandwidth demand can be adjusted by selecting a different encoding mode.

The number of frames encapsulated in each RTP payload highly influences the overall bandwidth of the RTP stream due to header overhead constraints. Packetizing more frames in each RTP payload can reduce the number of packets sent and hence the overhead from IP/UDP/RTP headers, at the expense of increased delay.

If in-band redundancy scheme is used to protect against packet loss, the amount of introduced redundancy will need to be regulated so that the use of redundancy itself does not cause a congestion problem. In other words, a sender SHALL NOT increase the total bitrate when adding redundancy in response to packet loss, and needs instead to adjust it down in accordance to the congestion control algorithm being run. Thus, when adding redundancy, the media bitrate will need to be reduced to provide room for the redundancy.

8. IANA Considerations

One media type has been defined and needs registration in the media types registry.

9. Normative References

[RFC 2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC 3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC 3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC 4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC 3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K., "The Secure Real-Time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC 5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC 4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

10. Author(s) Information:

Sergey Ikonin
email: info@spiritdsp.com

Russia 109004
Building 27, A. Solzhenitsyna street
Tel: +7 495 661-2178
Fax: +7 495 912-6786

11. Disclaimer

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. **The person(s) controlling the copyright in some of this material** may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

12. Legal Terms

APPENDIX A. RETRIEVING FRAME INFORMATION

This appendix contains the c-code for implementation of frame parsing function. This function extracts information about coded frame including frame size, number of layers, size of each layer and size of perceptual sensitive classes.

A.1. get_frame_info.c

```
/*
  Copyright (c) 2010 IETF Trust and the persons identified as authors
  of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

  - Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

  - Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

  - Neither the name of the Xiph.org Foundation nor the names of its
    contributors may be used to endorse or promote products derived from
    this software without specific prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
  A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE
  FOUNDATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
  BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
  OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
  AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
  THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
  DAMAGE.
*/

/*****************************************************************

  get_frame_info.c

  Retrieving frame information for IP-MR Speech Codec
```

```
*****************************************************************/

#define RATES_NUM       6   // number of codec rates
#define SENSE_CLASSES   6   // number of sensitivity classes (A..F)

// frame types
#define FT_SPEECH       0   // active speech
#define FT_DTX_SID      1   // silence insertion descriptor

// get specified bit from coded data
int GetBit(unsigned char *data, int curBit)
{
  return ((data[curBit >> 3] >> (curBit % 8)) & 1);
}

// retrieve frame information
int GetFrameInfo(           // o: frame size in bits
  short rate,               // i: encoding rate (0..5)
  short base_rate,          // i: base (core) layer rate,
                            //    if base_rate > rate, then assumed
                            //    that base_rate = rate.
  unsigned char *pCoded,    // i: coded bit frame
  short pLayerBits          // o: number of bits in layers
      [RATES_NUM],
  short pSenseBits          // o: number of bits in sensitivity classes
      [SENSE_CLASSES],
  short *nLayers            // o: number of layers
)
{
  static const short Bits_1[4]    = {0, 9, 9, 15};
  static const short Bits_2[16]   = { 43,50,36,31,46,48,40,44,47,43,44,
                                      45,43,44,47,36};
```

```
  static const short Bits_3[2][6] = {{13, 11, 23, 33, 36, 31},
                                     {25,  0, 23, 32, 36, 31},};

  int FrType;
  int i,nBits;

  if (rate < 0 || rate > 5) {
    return 0; // incorrect stream
  }

  for(i = 0; i < SENSE_CLASSES; i++) {
    pSenseBits[i] = 0;
  }

  nBits = 0;
  // extract frame type bit if required
  FrType = GetBit(pCoded, nBits++) ? FT_SPEECH : FT_DTX_SID;

  {
    int cw_0;
    int b[14];

    // extract meaning bits
    for(i = 0 ; i < 14; i++) {
        b[i] = GetBit(pCoded, nBits++);
    }

    // parse
    if(FrType == FT_DTX_SID) {
      cw_0 = (b[0]<<0)|(b[1]<<1)|(b[2]<<2)|(b[3]<<3);
      rate = 0;
      pSenseBits[0] = 10 + Bits_2[cw_0];
    } else {

      int i, idx;
      int nFlag_1, nFlag_2, cw_1, cw_2;

      nFlag_1 = b[0] + b[2] + b[4] + b[6];
      cw_1 = (cw_1 << 1) | b[0];
      cw_1 = (cw_1 << 1) | b[2];
      cw_1 = (cw_1 << 1) | b[4];
      cw_1 = (cw_1 << 1) | b[6];

      nFlag_2 = b[1] + b[3] + b[5] + b[7];
      cw_2 = (cw_2 << 1) | b[1];
      cw_2 = (cw_2 << 1) | b[3];
      cw_2 = (cw_2 << 1) | b[5];
```

```
      cw_2 = (cw_2 << 1) | b[7];

      cw_0 = (b[10]<<0)|(b[11]<<1)|(b[12]<<2)|(b[13]<<3);
      if (base_rate < 0)    base_rate = 0;
      if (base_rate > rate) base_rate = rate;
      idx = base_rate == 0 ? 0 : 1;

      pSenseBits[0] = 15+Bits_2[cw_0];
      pSenseBits[1] = Bits_1[(cw_1 >> 0)&0x3] + Bits_1[(cw_1>>2)&0x3];
      pSenseBits[2] = nFlag_1*5;
      pSenseBits[3] = nFlag_2*30;
      pSenseBits[5] = (4 - nFlag_2)*(Bits_3[idx][0]);

      for (i = 1; i < rate+1; i++) {
        pLayerBits[i] = 4*(Bits_3[idx][i]);
      }
    }

    pLayerBits[0] = 0;
    for (i = 0; i < SENSE_CLASSES; i++) {
        pLayerBits[0] += pSenseBits[i];
    }

    *nLayers = rate+1;
  }

  {
    // count total frame size
    int payloadBitCount = 0;
    for (i = 0; i < *nLayers; i++) {
      payloadBitCount += pLayerBits[i];
    }
    return payloadBitCount;
  }
}
```

Authors' Addresses

SPIRIT DSP
Building 27, A. Solzhenitsyna street
109004, Moscow, RUSSIA

Tel: +7 495 661-2178
Fax: +7 495 912-6786
EMail: info@spiritdsp.com