RTP Payload Format for Flexible Forward Error Correction (FEC)
draft-ietf-payload-flexible-fec-scheme-03

Abstract

   This document defines new RTP payload formats for the Forward Error
   Correction (FEC) packets that are generated by the non-interleaved
   and interleaved parity codes from a source media encapsulated in RTP.
   These parity codes are systematic codes, where a number of repair
   symbols are generated from a set of source symbols. These repair
   symbols are sent in a repair flow separate from the source flow that
   carries the source symbols. The non-interleaved and interleaved
   parity codes which are defined in this specification offer a good
   protection against random and bursty packet losses, respectively, at
   a cost of decent complexity. Moreover, alternate FEC codes may be
   used with the payload formats presented. The RTP payload formats
   that are defined in this document address the scalability issues
   experienced with the earlier specifications including RFC 2733, RFC
   5109 and SMPTE 2022-1, and offer several improvements. Due to these
   changes, the new payload formats are not backward compatible with the
   earlier specifications, but endpoints that do not implement the
   scheme can still work by simply ignoring the FEC packets.

Status of This Memo

This Internet-Draft will expire on May 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

   This document defines new RTP payload formats for the Forward Error
   Correction (FEC) that is generated by the non-interleaved and
   interleaved parity codes from a source media encapsulated in RTP
   [RFC3550]. These payload formats may also be used for other types of
   FEC codes. The type of the source media protected by these parity
   codes can be audio, video, text or application. The FEC data are
   generated according to the media type parameters, which are
   communicated out-of-band (e.g., in SDP). Furthermore, the
   associations or relationships between the source and repair flows may
   be communicated in-band or out-of-band. Situations where adaptivitiy
   of FEC parameters is desired, the endpoint can use the in-band
   mechanism, whereas when the FEC parameters are fixed, the endpoint
   may prefer to negotiate them out-of-band.

   The repair packets proposed in this document protect the source
   stream packets that belong to the same RTP session.

1.1. Parity Codes

   Both the non-interleaved and interleaved parity codes use the
   eXclusive OR (XOR) operation to generate the repair symbols. In a
   nutshell, the following steps take place:

1. The sender determines a set of source packets to be protected by
    FEC based on the media type parameters.

2. The sender applies the XOR operation on the source symbols to
    generate the required number of repair symbols.

3. The sender packetizes the repair symbols and sends the repair
    packet(s) along with the source packets to the receiver(s) (in
    different flows). The repair packets may be sent proactively or
    on-demand.

Note that the source and repair packets belong to different source
and repair flows, and the sender must provide a way for the receivers
to demultiplex them, even in the case they are sent in the same
5-tuple (i.e., same source/destination address/port with UDP). This
is required to offer backward compatibility for endpoints that do not
understand the FEC packets (See Section 4). At the receiver side, if
all of the source packets are successfully received, there is no need
for FEC recovery and the repair packets are discarded. However, if
there are missing source packets, the repair packets can be used to
recover the missing information. Figure 1 and Figure 2 describe
example block diagrams for the systematic parity FEC encoder and
decoder, respectively.

```
                           +------------+
+--+  +--+  +--+  +--+ --> | Systematic | --> +--+  +--+   +--+   +--+
+--+  +--+  +--+  +--+     | Parity FEC |     +--+  +--+   +--+   +--+
                          |  Encoder   |
                          |  (Sender)  | --> +==+  +==+
                          +------------+     +==+  +==+

Source Packet: +--+    Repair Packet: +==+
               +--+                    +==+
```
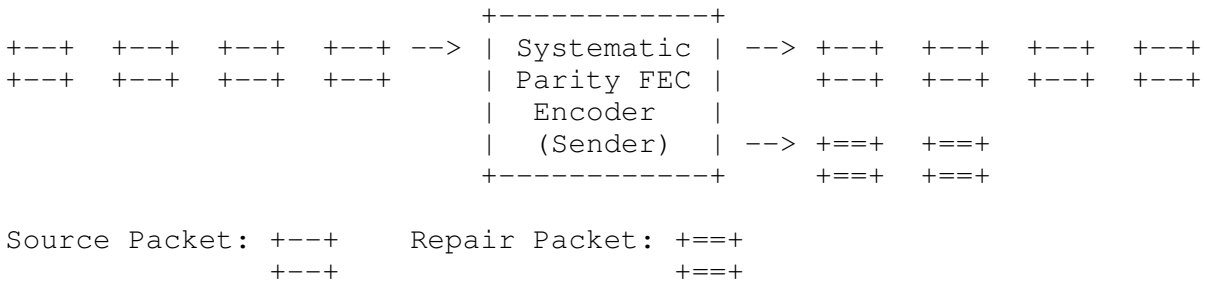
        Figure 1: Block diagram for systematic parity FEC encoder

```
                           +------------+
+--+     X     X    +--+ --> | Systematic | --> +--+  +--+   +--+   +--+
+--+               +--+     | Parity FEC |     +--+  +--+   +--+   +--+
                          |  Decoder   |
              +==+  +==+ --> | (Receiver) |
              +==+  +==+     +------------+

Source Packet: +--+    Repair Packet: +==+    Lost Packet: X
               +--+                    +==+
```
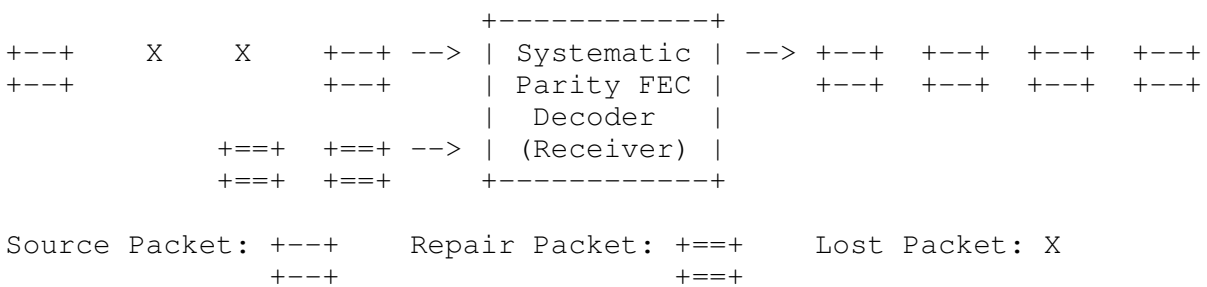
        Figure 2: Block diagram for systematic parity FEC decoder

In Figure 2, it is clear that the FEC packets have to be received by
the endpoint within a certain amount of time for the FEC recovery
process to be useful. In this document, we refer to the time that
spans a FEC block, which consists of the source packets and the
corresponding repair packets, as the repair window. At the receiver
side, the FEC decoder should wait at least for the duration of the
repair window after getting the first packet in a FEC block, to allow
all the repair packets to arrive. (The waiting time can be adjusted
if there are missing packets at the beginning of the FEC block.) The
FEC decoder can start decoding the already received packets sooner;
however, it should not register a FEC decoding failure until it waits
at least for the duration of the repair window.

Suppose that we have a group of D x L source packets that have
sequence numbers starting from 1 running to D x L, and a repair
packet is generated by applying the XOR operation to every L
consecutive packets as sketched in Figure 3. This process is
referred to as 1-D non-interleaved FEC protection. As a result of
this process, D repair packets are generated, which we refer to as
non-interleaved (or row) FEC packets.

```
+---------------------------------------------------+   ---    +===+
| S_1          S_2          S3          ... S_L     | + |XOR| = |R_1|
+---------------------------------------------------+   ---    +===+
+---------------------------------------------------+   ---    +===+
| S_L+1        S_L+2        S_L+3        ... S_2xL   | + |XOR| = |R_2|
+---------------------------------------------------+   ---    +===+
    .             .            .              .         .         .
    .             .            .              .         .         .
    .             .            .              .         .         .
+---------------------------------------------------+   ---    +===+
| S_(D-1)xL+1  S_(D-1)xL+2  S_(D-1)xL+3 ...  S_DxL  | + |XOR| = |R_D|
+---------------------------------------------------+   ---    +===+
```
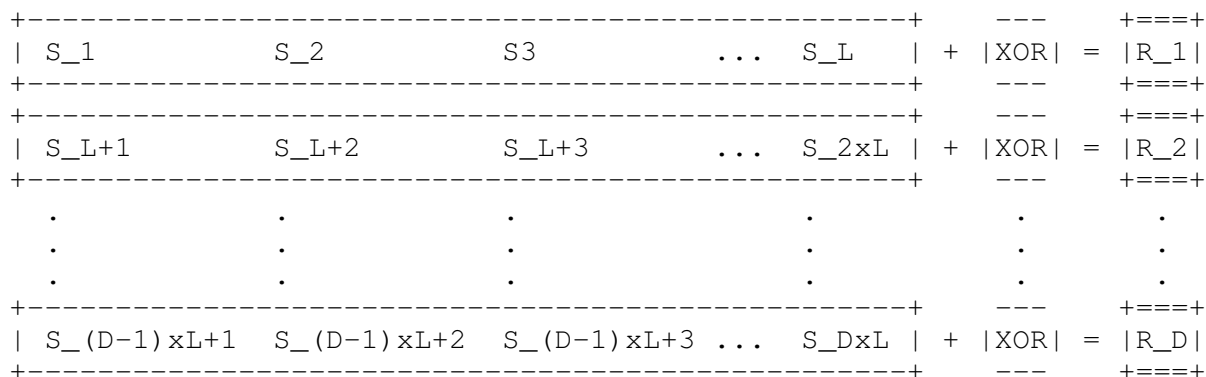
            Figure 3: Generating non-interleaved (row) FEC packets

If we apply the XOR operation to the group of the source packets
whose sequence numbers are L apart from each other, as sketched in
Figure 4. In this case the endpoint generates L repair packets.
This process is referred to as 1-D interleaved FEC protection, and
the resulting L repair packets are referred to as interleaved (or
column) FEC packets.

```
  +------------+ +------------+ +------------+    +-------+
  | S_1        | | S_2        | | S3         | ...| S_L   |
  | S_L+1      | | S_L+2      | | S_L+3      | ...| S_2xL |
  | .          | | .          | | .          |    |       |
  | .          | | .          | | .          |    |       |
  | .          | | .          | | .          |    |       |
  | S_(D-1)xL+1| | S_(D-1)xL+2| | S_(D-1)xL+3| ...| S_DxL |
  +------------+ +------------+ +------------+    +-------+
       +              +              +               +
  ------------   ------------   ------------      -------
  |    XOR    |  |    XOR    |  |    XOR    | ...|  XOR  |
  ------------   ------------   ------------      -------
       =              =              =               =
     +===+          +===+          +===+           +===+
     |C_1|          |C_2|          |C_3|    ...    |C_L|
     +===+          +===+          +===+           +===+
```
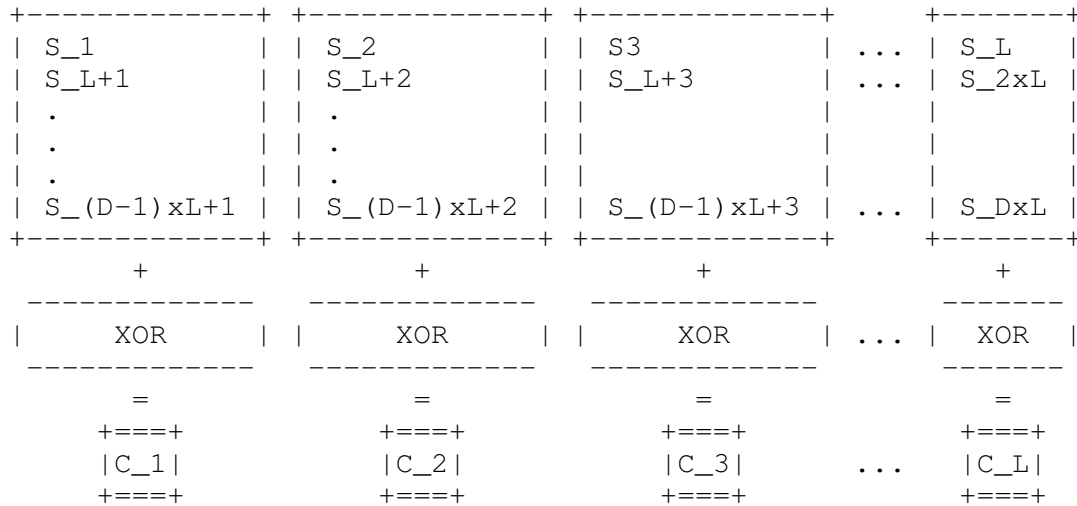
       Figure 4: Generating interleaved (column) FEC packets

1.1.1. Use Cases for 1-D FEC Protection

   We generate one non-interleaved repair packet out of L consecutive
   source packets or one interleaved repair packet out of D non-
   consecutive source packets. Regardless of whether the repair packet
   is a non-interleaved or an interleaved one, it can provide a full
   recovery of the missing information if there is only one packet
   missing among the corresponding source packets. This implies that
   1-D non-interleaved FEC protection performs better when the source
   packets are randomly lost. However, if the packet losses occur in
   bursts, 1-D interleaved FEC protection performs better provided that
   L is chosen large enough, i.e., L-packet duration is not shorter than
   the observed burst duration. If the sender generates non-interleaved
   FEC packets and a burst loss hits the source packets, the repair
   operation fails. This is illustrated in Figure 5.

```
+---+                   +---+  +===+
| 1 |     X       X     | 4 |  |R_1|
+---+                   +---+  +===+


+---+  +---+  +---+  +---+  +===+
| 5 |  | 6 |  | 7 |  | 8 |  |R_2|
+---+  +---+  +---+  +---+  +===+


+---+  +---+  +---+  +---+  +===+
| 9 |  | 10|  | 11|  | 12|  |R_3|
+---+  +---+  +---+  +---+  +===+
```
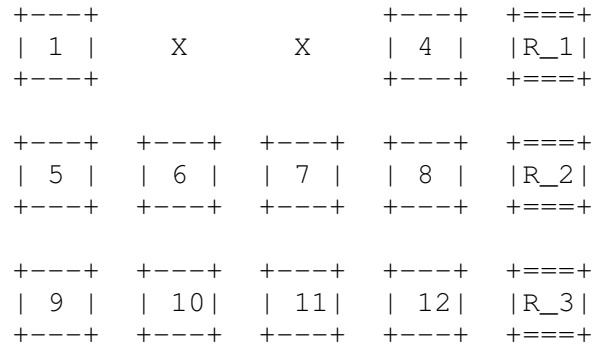
         Figure 5: Example scenario where 1-D non-interleaved FEC protection
                      fails error recovery (Burst Loss)

   The sender may generate interleaved FEC packets to combat with the
   bursty packet losses. However, two or more random packet losses may
   hit the source and repair packets in the same column. In that case,
   the repair operation fails as well. This is illustrated in Figure 6.
   Note that it is possible that two burst losses may occur back-to-
   back, in which case interleaved FEC packets may still fail to recover
   the lost data.

```
+---+                   +---+  +---+
| 1 |     X       | 3 |  | 4 |
+---+                   +---+  +---+


+---+                   +---+  +---+
| 5 |     X       | 7 |  | 8 |
+---+                   +---+  +---+


+---+  +---+  +---+  +---+
| 9 |  | 10|  | 11|  | 12|
+---+  +---+  +---+  +---+


+===+  +===+  +===+  +===+
|C_1|  |C_2|  |C_3|  |C_4|
+===+  +===+  +===+  +===+
```
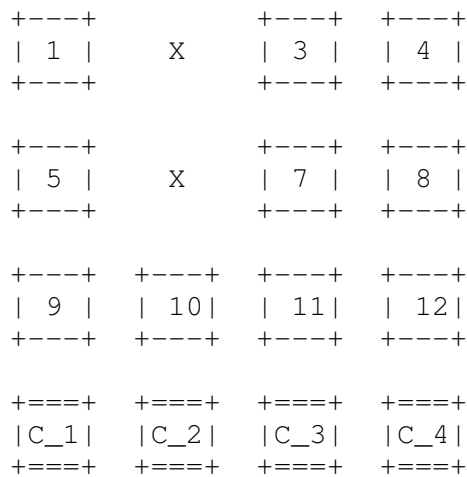
      Figure 6: Example scenario where 1-D interleaved FEC protection fails
                      error recovery (Periodic Loss)

1.1.2. Use Cases for 2-D Parity FEC Protection

   In networks where the source packets are lost both randomly and in
   bursts, the sender ought to generate both non-interleaved and
   interleaved FEC packets. This type of FEC protection is known as 2-D
   parity FEC protection. At the expense of generating more FEC
   packets, thus increasing the FEC overhead, 2-D FEC provides superior
   protection against mixed loss patterns. However, it is still
   possible for 2-D parity FEC protection to fail to recover all of the
   lost source packets if a particular loss pattern occurs. An example
   scenario is illustrated in Figure 7.

```
            +---+                   +---+  +===+
            | 1 |      X        X   | 4 |  |R_1|
            +---+                   +---+  +===+

            +---+  +---+  +---+  +---+  +===+
            | 5 |  | 6 |  | 7 |  | 8 |  |R_2|
            +---+  +---+  +---+  +---+  +===+

            +---+                   +---+  +===+
            | 9 |      X        X   | 12|  |R_3|
            +---+                   +---+  +===+

            +===+  +===+  +===+  +===+
            |C_1|  |C_2|  |C_3|  |C_4|
            +===+  +===+  +===+  +===+
```
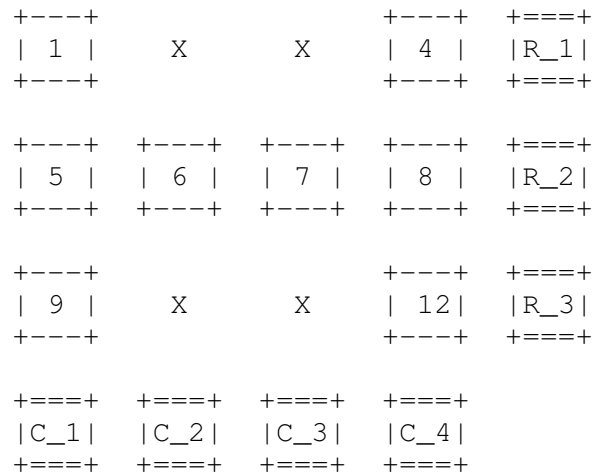
        Figure 7: Example scenario #1 where 2-D parity FEC protection fails
                             error recovery

   2-D parity FEC protection also fails when at least two rows are
   missing a source and the FEC packet and the missing source packets
   (in at least two rows) are aligned in the same column. An example
   loss pattern is sketched in Figure 8. Similarly, 2-D parity FEC
   protection cannot repair all missing source packets when at least two
   columns are missing a source and the FEC packet and the missing
   source packets (in at least two columns) are aligned in the same row.

```
        +---+  +---+           +---+
        | 1 |  | 2 |    X      | 4 |     X
        +---+  +---+           +---+


        +---+  +---+  +---+  +---+  +===+
        | 5 |  | 6 |  | 7 |  | 8 |  |R_2|
        +---+  +---+  +---+  +---+  +===+


        +---+  +---+           +---+
        | 9 |  | 10|    X      | 12|     X
        +---+  +---+           +---+


        +===+  +===+  +===+  +===+
        |C_1|  |C_2|  |C_3|  |C_4|
        +===+  +===+  +===+  +===+
```
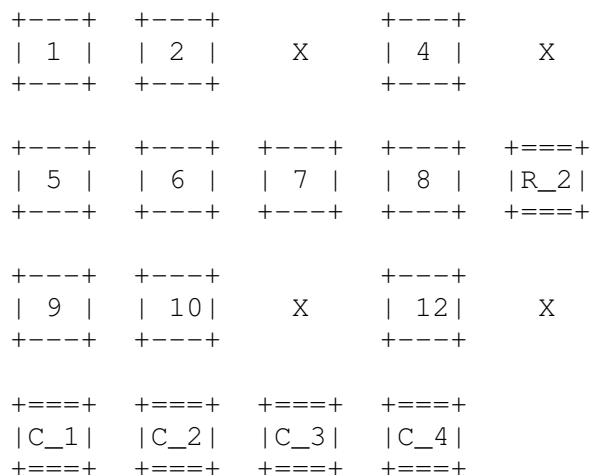
         Figure 8: Example scenario #2 where 2-D parity FEC protection fails
                              error recovery

1.1.3. Overhead Computation

   The overhead is defined as the ratio of the number of bytes belonging
   to the repair packets to the number of bytes belonging to the
   protected source packets.

   Generally, repair packets are larger in size compared to the source
   packets. Also, not all the source packets are necessarily equal in
   size. However, if we assume that each repair packet carries an equal
   number of bytes carried by a source packet, we can compute the
   overhead for different FEC protection methods as follows:

   o 1-D Non-interleaved FEC Protection: Overhead = $1/L$

   o 1-D Interleaved FEC Protection: Overhead = $1/D$

   o 2-D Parity FEC Protection: Overhead = $1/L + 1/D$

   where L and D are the number of columns and rows in the source block,
   respectively.

2. Requirements Notation

   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 [RFC2119].

3. Definitions and Notations

3.1. Definitions

   This document uses a number of definitions from [RFC6363].

3.2. Notations

   o L: Number of columns of the source block.

   o D: Number of rows of the source block.

   o bitmask: Run-length encoding of packets protected by a FEC packet.
      If the bit i in the mask is set to 1, the source packet number N +
      i is protected by this FEC packet. Here, N is the sequence number
      base, which is indicated in the FEC packet as well.

4. Packet Formats

   This section defines the formats of the source and repair packets.

4.1. Source Packets

   The source packets MUST contain the information that identifies the
   source block and the position within the source block occupied by the
   packet. Since the source packets that are carried within an RTP
   stream already contain unique sequence numbers in their RTP headers
   [RFC3550], we can identify the source packets in a straightforward
   manner and there is no need to append additional field(s). The
   primary advantage of not modifying the source packets in any way is
   that it provides backward compatibility for the receivers that do not
   support FEC at all. In multicast scenarios, this backward
   compatibility becomes quite useful as it allows the non-FEC-capable
   and FEC-capable receivers to receive and interpret the same source
   packets sent in the same multicast session.

4.2. Repair Packets

   The repair packets MUST contain information that identifies the
   source block they pertain to and the relationship between the
   contained repair symbols and the original source block. For this
   purpose, we use the RTP header of the repair packets as well as
   another header within the RTP payload, which we refer to as the FEC
   header, as shown in Figure 9.

   Note that all the source stream packets that are protected by a
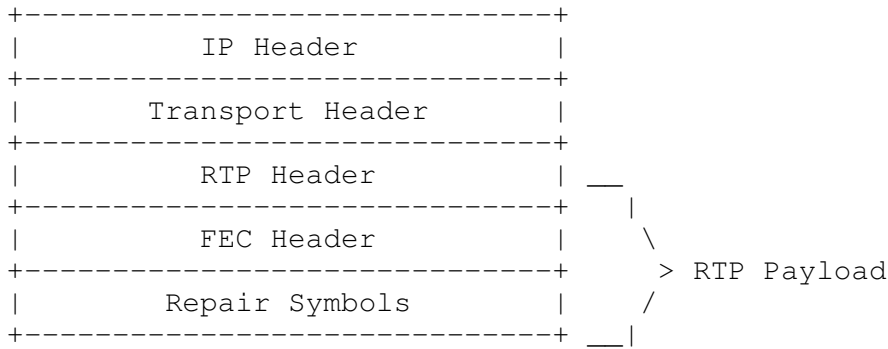   particular FEC packet need to be in the same RTP session.

```
        +----------------------------+
        |          IP Header         |
        +----------------------------+
        |      Transport Header      |
        +----------------------------+
        |         RTP Header         | __
        +----------------------------+   |
        |         FEC Header         |    \
        +----------------------------+     > RTP Payload
        |       Repair Symbols       |    /
        +----------------------------+ __|
```

                Figure 9: Format of repair packets

   The RTP header is formatted according to [RFC3550] with some further
   clarifications listed below:

   o Marker (M) Bit: This bit is not used for this payload type, and
      SHALL be set to 0.

   o Payload Type: The (dynamic) payload type for the repair packets is
      determined through out-of-band means. Note that this document
      registers new payload formats for the repair packets (Refer to
      Section 5 for details). According to [RFC3550], an RTP receiver
      that cannot recognize a payload type must discard it. This
      provides backward compatibility. If a non-FEC-capable receiver
      receives a repair packet, it will not recognize the payload type,
      and hence, will discard the repair packet.

   o Sequence Number (SN): The sequence number has the standard
      definition. It MUST be one higher than the sequence number in the
      previously transmitted repair packet. The initial value of the
      sequence number SHOULD be random (unpredictable, based on
      [RFC3550]).

   o Timestamp (TS): The timestamp SHALL be set to a time corresponding
      to the repair packet's transmission time. Note that the timestamp
      value has no use in the actual FEC protection process and is
      usually useful for jitter calculations.

   o Synchronization Source (SSRC): The SSRC value SHALL be randomly
      assigned as suggested by [RFC3550]. This allows the sender to
      multiplex the source and repair flows on the same port, or
      multiplex multiple repair flows on a single port. The repair
      flows SHOULD use the RTCP CNAME field to associate themselves with
      the source flow.

In some networks, the RTP Source, which produces the source
packets and the FEC Source, which generates the repair packets
from the source packets may not be the same host. In such
scenarios, using the same CNAME for the source and repair flows
means that the RTP Source and the FEC Source MUST share the same
CNAME (for this specific source-repair flow association). A
common CNAME may be produced based on an algorithm that is known
both to the RTP and FEC Source [RFC7022]. This usage is compliant
with [RFC3550].

Note that due to the randomness of the SSRC assignments, there is
a possibility of SSRC collision. In such cases, the collisions
MUST be resolved as described in [RFC3550].

The format of the FEC header is shown in Figure 10.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|F| P|X|  CC   |M| PT recovery |          length recovery      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          TS recovery                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SSRCCount   |                   reserved                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            SSRC_i                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         SN base_i          |k|          Mask [0-14]           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|k|               Mask [15-45] (optional)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|k|                                                             |
+-+               Mask [46-108] (optional)                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    ... next in SSRC_i ...                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
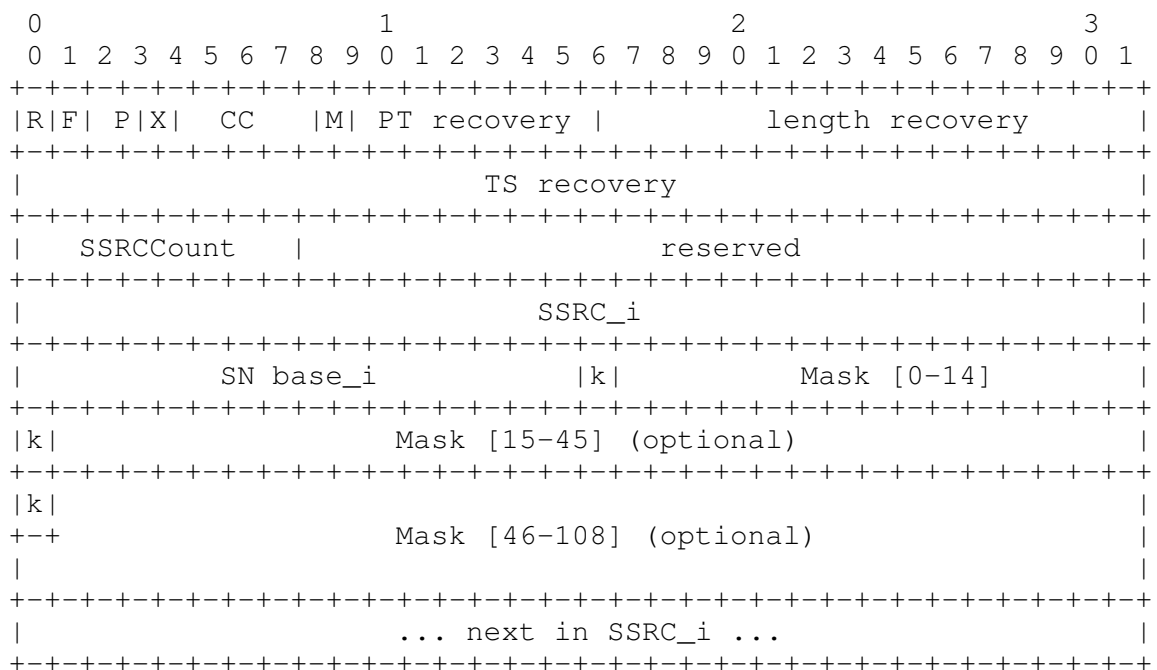
Figure 10: Format of the FEC header

The FEC header consists of the following fields:

o The R bit MUST be set to 1 to indicate a retransmission packet,
   and MUST be set to 0 for repair packets.

o The F field (1 bit) indicates the type of the mask. Namely:

```
 +--------------+-----------------------------------+
 |    F bit     | Use                               |
 +--------------+-----------------------------------+
 |      0       | flexible mask                     |
 |      1       | packets indicated by offset M and N |
 +--------------+-----------------------------------+
```

Figure 11: F-bit values

o The P, X, CC, M and PT recovery fields are used to determine the
   corresponding fields of the recovered packets.

o The Length recovery (16 bits) field is used to determine the
   length of the recovered packets.

o The TS recovery (32 bits) field is used to determine the timestamp
   of the recovered packets.

o The SSRC count (8 bits) field describes the number of SSRCs
   protected by the FEC packet. 0 is not a valid value, and the
   packet MUST be ignored.

o The Reserved (24 bits) field are reserved for future use. It MUST
   be set to zero by senders and ignored by receivers (see [RFC6709],
   Section 4.2).

o The SSRC_i (32 bits) field describes the SSRC of the packets
   protected by this particular FEC packet. If a FEC packet contains
   protects multiple SSRCs (indicated by the SSRC Count > 1), there
   will be multiple blocks of data containing the SSRC, SN base and
   Mask fields.

o The SN base_i (16 bits) field indicates the lowest sequence
   number, taking wrap around into account, of the source packets for
   a particular SSSRC (indicated in SSRC_i) protected by this repair
   packet.

o If the F-bit is set to 0, it represents that the source packets of
   all the SSRCs protected by this particular repair packet are
   indicated by using a flexible bitmask. Mask is a run-length
   encoding of packets for a particular SSRC_i protected by the FEC
   packet. Where a bit j set to 1 indicates that the source packet
   with sequence number (SN base_i + j + 1) is protected by this FEC
   packet.

o The k-bit in the bitmasks indicates if it is 15-, 46-, or a
   109-bitmask. k=0 denotes that there is one more k-bit set, and
   k=1 denotes that it is the last block of bit mask. While parsing
   the header, the current count of the number of k-bit gives the
   size of the bit mask v = count(k). Size of next bitmask =
   $2^{(v+3)}-1$.

o Editor's note: If we limit the number of k-bits to 3, we could
   essentially remove the last k-bit.

o

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0| P|X|  CC  |M| PT recovery |         length recovery        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           TS recovery                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SSRCCount   |                  reserved                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            SSRC_i                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          SN base_i          |k|        Mask [0-14]             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|k|             Mask [15-45] (optional)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|k|                                                              |
+-+                 Mask [46-108] (optional)                     |
|                                                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    ... next in SSRC_i ...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
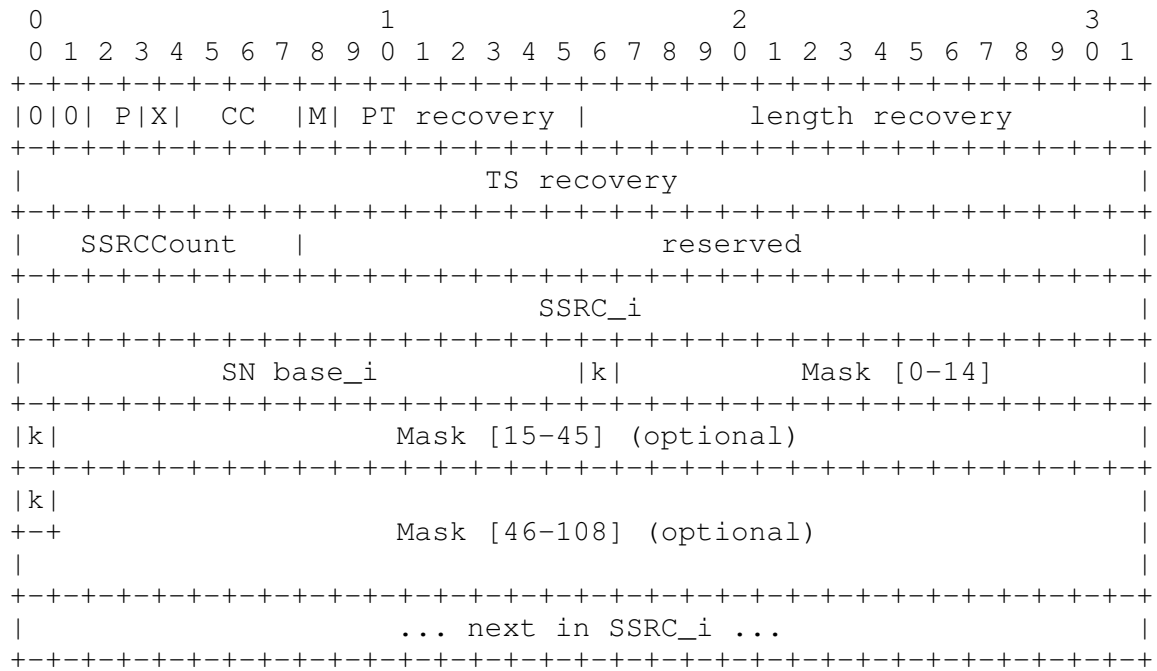
Figure 12: Protocol format for F=0

o If the F-bit is set to 1, it represents that the source packets of
   all the SSRCs protected by this particular repair packet are
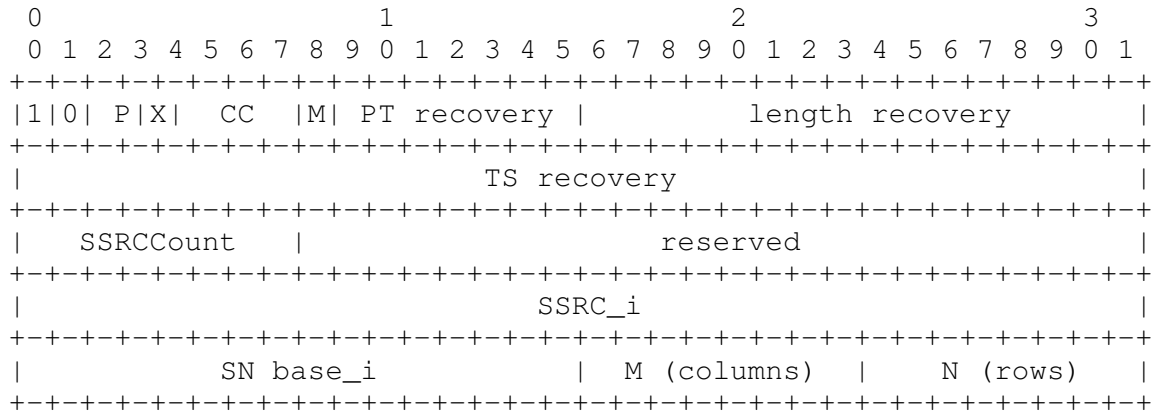   indicated by using fixed offsets.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|0| P|X|  CC   |M| PT recovery |          length recovery     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          TS recovery                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SSRCCount    |                  reserved                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SSRC_i                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          SN base_i            |  M (columns)  |   N (rows)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Protocol format for F=1

Consequently, the following conditions occur for M and N values:

If M>0, N=0, is Row FEC, and no column FEC will follow
         Hence, FEC = SN, SN+1, SN+2, ... , SN+(M-1), SN+M.

If M>0, N=1, is Row FEC, and column FEC will follow.
            Hence, FEC = SN, SN+1, SN+2, ... , SN+(M-1), SN+M.
       and more to come

If M>0, N>1, indicates column FEC of every M packet
            in a group of N packets starting at SN base.
            Hence, FEC = SN+(Mx0), SN+(Mx1), ... , SN+(MxN).

Figure 14: Interpreting the M and N field values

By setting R to 1, F to 1, this FEC protects only one packet, i.e.,
the FEC payload carries just the packet indicated by SN Base_i, which
is effectively retransmitting the packet.

Note that the parsing of this packet is different. The sequence
number (SN base_i) replaces the length recovery in the FEC packet.
The SSRC_count which would be 1, M and N would be set to 0, and the
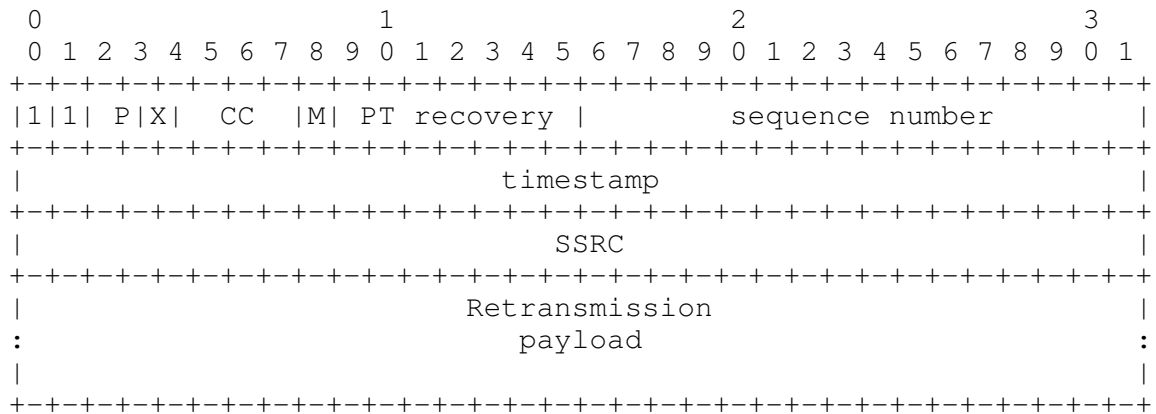reserved bits from the FEC header are removed. By doing this, we
save 64 bits.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|1| P|X|  CC   |M| PT recovery |           sequence number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             SSRC                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Retransmission                        |
:                            payload                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Protocol format for Retransmission

The details on setting the fields in the FEC header are provided in Section 6.2.

It should be noted that a mask-based approach (similar to the ones specified in [RFC2733] and [RFC5109]) may not be very efficient to indicate which source packets in the current source block are associated with a given repair packet. In particular, for the applications that would like to use large source block sizes, the size of the mask that is required to describe the source-repair packet associations may be prohibitively large. The 8-bit fields proposed in [SMPTE2022-1] indicate a systematized approach. Instead the approach in this document uses the 8-bit fields to indicate packet offsets protected by the FEC packet. The approach in [SMPTE2022-1] is inherently more efficient for regular patterns, it does not provide flexibility to represent other protection patterns (e.g., staircase).

5. Payload Format Parameters

   This section provides the media subtype registration for the non-interleaved and interleaved parity FEC. The parameters that are required to configure the FEC encoding and decoding operations are also defined in this section. If no specific FEC code is specified in the subtype, then the FEC code defaults to the parity code defined in this specification.

5.1. Media Type Registration - Parity Codes

   This registration is done using the template defined in [RFC6838] and following the guidance provided in [RFC3555].

Note to the RFC Editor: In the following sections, please replace "XXXX" with the number of this document prior to publication as an RFC.

5.1.1. Registration of audio/flexfec

   Type name: audio

   Subtype name: flexfec

   Required parameters:

   o rate: The RTP timestamp (clock) rate. The rate SHALL be larger
      than 1000 Hz to provide sufficient resolution to RTCP operations.
      However, it is RECOMMENDED to select the rate that matches the
      rate of the protected source RTP stream.

   o repair-window: The time that spans the source packets and the
      corresponding repair packets. The size of the repair window is
      specified in microseconds.

   Optional parameters:

   o L: indicates the number of columns of the source block that are
      protected by this FEC block and it applies to all the source
      SSRCs. L is a positive integer.

   o D: indicates the number of rows of the source block that are
      protected by this FEC block and it applies to all the source
      SSRCs. D is a positive integer.

   o ToP: indicates the type of protection applied by the sender: 0 for
      1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC
      protection, and 2 for 2-D parity FEC protection. The ToP value of
      3 is reserved for future uses.

   Encoding considerations: This media type is framed (See Section 4.8
   in the template document [RFC6838]) and contains binary data.

   Security considerations: See Section 9 of [RFCXXXX].

   Interoperability considerations: None.

   Published specification: [RFCXXXX].

   Applications that use this media type: Multimedia applications that
   want to improve resiliency against packet loss by sending redundant
   data in addition to the source media.

   Fragment identifier considerations: None.

   Additional information: None.

   Person & email address to contact for further information: Varun
   Singh <varun@callstats.io> and IETF Audio/Video Transport Payloads
   Working Group.

   Intended usage: COMMON.

   Restriction on usage: This media type depends on RTP framing, and
   hence, is only defined for transport via RTP [RFC3550].

   Author: Varun Singh <varun@callstats.io>.

   Change controller: IETF Audio/Video Transport Working Group delegated
   from the IESG.

   Provisional registration? (standards tree only): Yes.

5.1.2. Registration of video/flexfec

   Type name: video

   Subtype name: flexfec

   Required parameters:

   o rate: The RTP timestamp (clock) rate. The rate SHALL be larger
      than 1000 Hz to provide sufficient resolution to RTCP operations.
      However, it is RECOMMENDED to select the rate that matches the
      rate of the protected source RTP stream.

   o repair-window: The time that spans the source packets and the
      corresponding repair packets. The size of the repair window is
      specified in microseconds.

   Optional parameters:

   o L: indicates the number of columns of the source block that are
      protected by this FEC block and it applies to all the source
      SSRCs. L is a positive integer.

   o D: indicates the number of rows of the source block that are
      protected by this FEC block and it applies to all the source
      SSRCs. D is a positive integer.

   o ToP: indicates the type of protection applied by the sender: 0 for
      1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC
      protection, and 2 for 2-D parity FEC protection. The ToP value of
      3 is reserved for future uses.

   Encoding considerations: This media type is framed (See Section 4.8
   in the template document [RFC6838]) and contains binary data.

   Security considerations: See Section 9 of [RFCXXXX].

   Interoperability considerations: None.

   Published specification: [RFCXXXX].

   Applications that use this media type: Multimedia applications that
   want to improve resiliency against packet loss by sending redundant
   data in addition to the source media.

   Fragment identifier considerations: None.

   Additional information: None.

   Person & email address to contact for further information: Varun
   Singh <varun@callstats.io> and IETF Audio/Video Transport Payloads
   Working Group.

   Intended usage: COMMON.

   Restriction on usage: This media type depends on RTP framing, and
   hence, is only defined for transport via RTP [RFC3550].

   Author: Varun Singh <varun@callstats.io>.

   Change controller: IETF Audio/Video Transport Working Group delegated
   from the IESG.

   Provisional registration? (standards tree only): Yes.

5.1.3. Registration of text/flexfec

   Type name: text

   Subtype name: flexfec

   Required parameters:

   o rate: The RTP timestamp (clock) rate. The rate SHALL be larger
      than 1000 Hz to provide sufficient resolution to RTCP operations.

However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.

o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters:

o L: indicates the number of columns of the source block that are protected by this FEC block and it applies to all the source SSRCs. L is a positive integer.

o D: indicates the number of rows of the source block that are protected by this FEC block and it applies to all the source SSRCs. D is a positive integer.

o ToP: indicates the type of protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <vvarun@callstats.io> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun@callstats.io>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

5.1.4. Registration of application/flexfec

Type name: application

Subtype name: flexfec

Required parameters:

o   rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.

o   repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters:

o   L: indicates the number of columns of the source block that are protected by this FEC block and it applies to all the source SSRCs. L is a positive integer.

o   D: indicates the number of rows of the source block that are protected by this FEC block and it applies to all the source SSRCs. D is a positive integer.

o   ToP: indicates the type of protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun@callstats.io> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun@callstats.io>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

5.2. Mapping to SDP Parameters

   Applications that are using RTP transport commonly use Session Description Protocol (SDP) [RFC4566] to describe their RTP sessions. The information that is used to specify the media types in an RTP session has specific mappings to the fields in an SDP description. In this section, we provide these mappings for the media subtypes registered by this document. Note that if an application does not use SDP to describe the RTP sessions, an appropriate mapping must be defined and used to specify the media types and their parameters for the control/description protocol employed by the application.

   The mapping of the media type specification for "non-interleaved-parityfec" and "interleaved-parityfec" and their parameters in SDP is as follows:

   o The media type (e.g., "application") goes into the "m=" line as the media name.

   o The media subtype goes into the "a=rtpmap" line as the encoding name. The RTP clock rate parameter ("rate") also goes into the "a=rtpmap" line as the clock rate.

o The remaining required payload-format-specific parameters go into
   the "a=fmtp" line by copying them directly from the media type
   string as a semicolon-separated list of parameter=value pairs.

SDP examples are provided in Section 7.

5.2.1. Offer-Answer Model Considerations

When offering 1-D interleaved parity FEC over RTP using SDP in an
Offer/Answer model [RFC3264], the following considerations apply:

o Each combination of the L and D parameters produces a different
   FEC data and is not compatible with any other combination. A
   sender application may desire to offer multiple offers with
   different sets of L and D values as long as the parameter values
   are valid. The receiver SHOULD normally choose the offer that has
   a sufficient amount of interleaving. If multiple such offers
   exist, the receiver may choose the offer that has the lowest
   overhead or the one that requires the smallest amount of
   buffering. The selection depends on the application requirements.

o The value for the repair-window parameter depends on the L and D
   values and cannot be chosen arbitrarily. More specifically, L and
   D values determine the lower limit for the repair-window size.
   The upper limit of the repair-window size does not depend on the L
   and D values.

o Although combinations with the same L and D values but with
   different repair-window sizes produce the same FEC data, such
   combinations are still considered different offers. The size of
   the repair-window is related to the maximum delay between the
   transmission of a source packet and the associated repair packet.
   This directly impacts the buffering requirement on the receiver
   side and the receiver must consider this when choosing an offer.

o There are no optional format parameters defined for this payload.
   Any unknown option in the offer MUST be ignored and deleted from
   the answer. If FEC is not desired by the receiver, it can be
   deleted from the answer.

5.2.2. Declarative Considerations

In declarative usage, like SDP in the Real-time Streaming Protocol
(RTSP) [RFC2326] or the Session Announcement Protocol (SAP)
[RFC2974], the following considerations apply:

o   The payload format configuration parameters are all declarative
    and a participant MUST use the configuration that is provided for
    the session.

o   More than one configuration may be provided (if desired) by
    declaring multiple RTP payload types. In that case, the receivers
    should choose the repair flow that is best for them.

6.   Protection and Recovery Procedures - Parity Codes

   This section provides a complete specification of the 1-D and 2-D
   parity codes and their RTP payload formats.

6.1.   Overview

   The following sections specify the steps involved in generating the
   repair packets and reconstructing the missing source packets from the
   repair packets.

6.2.   Repair Packet Construction

   The RTP header of a repair packet is formed based on the guidelines
   given in Section 4.2.

   The FEC header includes 12 octets (or upto 28 octets when the longer
   optional masks are used). It is constructed by applying the XOR
   operation on the bit strings that are generated from the individual
   source packets protected by this particular repair packet. The set
   of the source packets that are associated with a given repair packet
   can be computed by the formula given in Section 6.3.1.

   The bit string is formed for each source packet by concatenating the
   following fields together in the order specified:

o   The first 64 bits of the RTP header (64 bits).

o   Unsigned network-ordered 16-bit representation of the source
    packet length in bytes minus 12 (for the fixed RTP header), i.e.,
    the sum of the lengths of all the following if present: the CSRC
    list, extension header, RTP payload and RTP padding (16 bits).

   By applying the parity operation on the bit strings produced from the
   source packets, we generate the FEC bit string. The FEC header is
   generated from the FEC bit string as follows:

o   The first (most significant) 2 bits in the FEC bit string are
    skipped. The MSK bits in the FEC header are set to the
    appropriate value, i.e., it depends on the chosen bitmask length.

o   The next bit in the FEC bit string is written into the P recovery
    bit in the FEC header.

o   The next bit in the FEC bit string is written into the X recovery
    bit in the FEC header.

o   The next 4 bits of the FEC bit string are written into the CC
    recovery field in the FEC header.

o   The next bit is written into the M recovery bit in the FEC header.

o   The next 7 bits of the FEC bit string are written into the PT
    recovery field in the FEC header.

o   The next 16 bits are skipped.

o   The next 32 bits of the FEC bit string are written into the TS
    recovery field in the FEC header.

o   The next 16 bits are written into the length recovery field in the
    FEC header.

o   Depending on the chosen MSK value, the bit mask of appropriate
    length will be set to the appropriate values.

As described in Section 4.2, the SN base field of the FEC header MUST
be set to the lowest sequence number of the source packets protected
by this repair packet. When MSK represents a bitmask (MSK=00,01,10),
the SN base field corresponds to the lowest sequence number indicated
in the bitmask. When MSK=11, the following considerations apply: 1)
for the interleaved FEC packets, this corresponds to the lowest
sequence number of the source packets that forms the column, 2) for
the non-interleaved FEC packets, the SN base field MUST be set to the
lowest sequence number of the source packets that forms the row.

The repair packet payload consists of the bits that are generated by
applying the XOR operation on the payloads of the source RTP packets.
If the payload lengths of the source packets are not equal, each
shorter packet MUST be padded to the length of the longest packet by
adding octet 0's at the end.

Due to this possible padding and mandatory FEC header, a repair
packet has a larger size than the source packets it protects. This
may cause problems if the resulting repair packet size exceeds the
Maximum Transmission Unit (MTU) size of the path over which the
repair flow is sent.

6.3. Source Packet Reconstruction

   This section describes the recovery procedures that are required to
   reconstruct the missing source packets. The recovery process has two
   steps. In the first step, the FEC decoder determines which source
   and repair packets should be used in order to recover a missing
   packet. In the second step, the decoder recovers the missing packet,
   which consists of an RTP header and RTP payload.

   In the following, we describe the RECOMMENDED algorithms for the
   first and second steps. Based on the implementation, different
   algorithms MAY be adopted. However, the end result MUST be identical
   to the one produced by the algorithms described below.

   Note that the same algorithms are used by the 1-D parity codes,
   regardless of whether the FEC protection is applied over a column or
   a row. The 2-D parity codes, on the other hand, usually require
   multiple iterations of the procedures described here. This iterative
   decoding algorithm is further explained in Section 6.3.4.

6.3.1. Associating the Source and Repair Packets

   We denote the set of the source packets associated with repair packet
   p* by set T(p*). Note that in a source block whose size is L columns
   by D rows, set T includes D source packets plus one repair packet for
   the FEC protection applied over a column, and L source packets plus
   one repair packet for the FEC protection applied over a row. Recall
   that 1-D interleaved and non-interleaved FEC protection can fully
   recover the missing information if there is only one source packet
   missing in set T. If there are more than one source packets missing
   in set T, 1-D FEC protection will not work.

6.3.1.1. Signaled in SDP

   The first step is associating the source and repair packets. If the
   endpoint relies entirely on out-of-band signaling (MSK=11, and
   M=N=0), then this information may be inferred from the media type
   parameters specified in the SDP description. Furthermore, the
   payload type field in the RTP header, assists the receiver
   distinguish an interleaved or non-interleaved FEC packet.

   Mathematically, for any received repair packet, p*, we can determine
   the sequence numbers of the source packets that are protected by this
   repair packet as follows:

$$p*\_snb + i * X\_1 \text{ (modulo 65536)}$$

where p*_snb denotes the value in the SN base field of p*'s FEC
header, X_1 is set to L and 1 for the interleaved and non-interleaved
FEC packets, respectively, and

$$0 <= i < X\_2$$

where X_2 is set to D and L for the interleaved and non-interleaved
FEC packets, respectively.

6.3.1.2. Using bitmasks

When using fixed size bitmasks (16-, 48-, 112-bits), the SN base
field in the FEC header indicates the lowest sequence number of the
source packets that forms the FEC packet. Finally, the bits maked by
"1" in the bitmask are offsets from the SN base and make up the rest
of the packets protected by the FEC packet. The bitmasks are able to
represent arbitrary protection patterns, for example, 1-D
interleaved, 1-D non-interleaved, 2-D, staircase.

6.3.1.3. Using M and N Offsets

When value of M is non-zero, the 8-bit fields indicate the offset of
packets protected by an interleaved (N>0) or non-interleaved (N=0)
FEC packet. Using a combination of interleaved and non-interleaved
FEC packets can form 2-D protection patterns.

Mathematically, for any received repair packet, p*, we can determine
the sequence numbers of the source packets that are protected by this
repair packet are as follows:

When N = 0:
  p*_snb, p*_snb+1,..., p*_snb+(M-1), p*_snb+M
When N > 0:
  p*_snb, p*_snb+(Mx1), p*_snb+(Mx2),..., p*_snb+(Mx(N-1)), p*_snb+(MxN)

6.3.2. Recovering the RTP Header

For a given set T, the procedure for the recovery of the RTP header
of the missing packet, whose sequence number is denoted by SEQNUM, is
as follows:

1.   For each of the source packets that are successfully received in
     T, compute the 80-bit string by concatenating the first 64 bits
     of their RTP header and the unsigned network-ordered 16-bit
     representation of their length in bytes minus 12.

2.   For the repair packet in T, compute the FEC bit string from the
     first 80 bits of the FEC header.

3.   Calculate the recovered bit string as the XOR of the bit strings
     generated from all source packets in T and the FEC bit string
     generated from the repair packet in T.

4.   Create a new packet with the standard 12-byte RTP header and no
     payload.

5.   Set the version of the new packet to 2. Skip the first 2 bits
     in the recovered bit string.

6.   Set the Padding bit in the new packet to the next bit in the
     recovered bit string.

7.   Set the Extension bit in the new packet to the next bit in the
     recovered bit string.

8.   Set the CC field to the next 4 bits in the recovered bit string.

9.   Set the Marker bit in the new packet to the next bit in the
     recovered bit string.

10. Set the Payload type in the new packet to the next 7 bits in the
     recovered bit string.

11. Set the SN field in the new packet to SEQNUM. Skip the next 16
     bits in the recovered bit string.

12. Set the TS field in the new packet to the next 32 bits in the
     recovered bit string.

13. Take the next 16 bits of the recovered bit string and set the
     new variable Y to whatever unsigned integer this represents
     (assuming network order). Convert Y to host order. Y
     represents the length of the new packet in bytes minus 12 (for
     the fixed RTP header), i.e., the sum of the lengths of all the
     following if present: the CSRC list, header extension, RTP
     payload and RTP padding.

14. Set the SSRC of the new packet to the SSRC of the source RTP
     stream.

This procedure recovers the header of an RTP packet up to (and
including) the SSRC field.

6.3.3. Recovering the RTP Payload

   Following the recovery of the RTP header, the procedure for the
   recovery of the RTP payload is as follows:

   1. Append Y bytes to the new packet.

   2. For each of the source packets that are successfully received in
       T, compute the bit string from the Y octets of data starting with
       the 13th octet of the packet. If any of the bit strings
       generated from the source packets has a length shorter than Y,
       pad them to that length. The padding of octet 0 MUST be added at
       the end of the bit string. Note that the information of the
       first 8 octets are protected by the FEC header.

   3. For the repair packet in T, compute the FEC bit string from the
       repair packet payload, i.e., the Y octets of data following the
       FEC header. Note that the FEC header may be 12, 16, 32 octets
       depending on the length of the bitmask.

   4. Calculate the recovered bit string as the XOR of the bit strings
       generated from all source packets in T and the FEC bit string
       generated from the repair packet in T.

   5. Append the recovered bit string (Y octets) to the new packet
       generated in Section 6.3.2.

6.3.4. Iterative Decoding Algorithm for the 2-D Parity FEC Protection

   In 2-D parity FEC protection, the sender generates both non-
   interleaved and interleaved FEC packets to combat with the mixed loss
   patterns (random and bursty). At the receiver side, these FEC
   packets are used iteratively to overcome the shortcomings of the 1-D
   non-interleaved/interleaved FEC protection and improve the chances of
   full error recovery.

   The iterative decoding algorithm runs as follows:

   1. Set num_recovered_until_this_iteration to zero

   2. Set num_recovered_so_far to zero

   3. Recover as many source packets as possible by using the non-
       interleaved FEC packets as outlined in Section 6.3.2 and
       Section 6.3.3, and increase the value of num_recovered_so_far by
       the number of recovered source packets.

4. Recover as many source packets as possible by using the
    interleaved FEC packets as outlined in Section 6.3.2 and
    Section 6.3.3, and increase the value of num_recovered_so_far by
    the number of recovered source packets.

5. If num_recovered_so_far > num_recovered_until_this_iteration
    ---num_recovered_until_this_iteration = num_recovered_so_far
    ---Go to step 3
    Else
    ---Terminate

The algorithm terminates either when all missing source packets are
fully recovered or when there are still remaining missing source
packets but the FEC packets are not able to recover any more source
packets. For the example scenarios when the 2-D parity FEC
protection fails full recovery, refer to Section 1.1.2. Upon
termination, variable num_recovered_so_far has a value equal to the
total number of recovered source packets.

Example:

Suppose that the receiver experienced the loss pattern sketched in
Figure 16.

```
                        +---+  +---+  +===+
            X       X   | 3 |  | 4 |  |R_1|
                        +---+  +---+  +===+

         +---+  +---+  +---+  +---+  +===+
         | 5 |  | 6 |  | 7 |  | 8 |  |R_2|
         +---+  +---+  +---+  +---+  +===+

         +---+                +---+  +===+
         | 9 |    X       X   | 12|  |R_3|
         +---+                +---+  +===+

         +===+  +===+  +===+  +===+
         |C_1|  |C_2|  |C_3|  |C_4|
         +===+  +===+  +===+  +===+
```
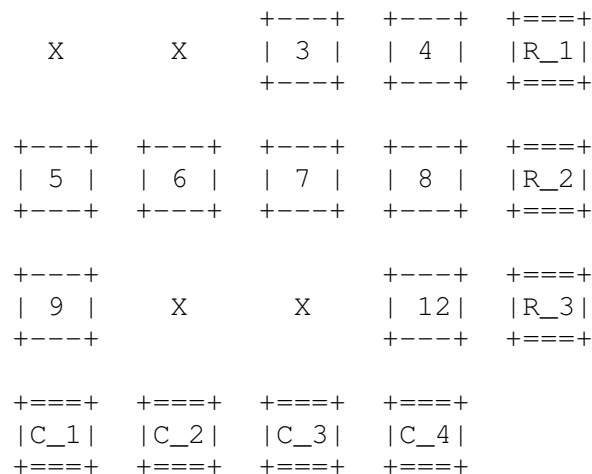
Figure 16: Example loss pattern for the iterative decoding algorithm

The receiver executes the iterative decoding algorithm and recovers
source packets #1 and #11 in the first iteration. The resulting
pattern is sketched in Figure 17.

```
          +---+            +---+  +---+  +===+
          | 1 |      X     | 3 |  | 4 |  |R_1|
          +---+            +---+  +---+  +===+


          +---+  +---+  +---+  +---+  +===+
          | 5 |  | 6 |  | 7 |  | 8 |  |R_2|
          +---+  +---+  +---+  +---+  +===+


          +---+            +---+  +---+  +===+
          | 9 |      X     | 11|  | 12|  |R_3|
          +---+            +---+  +---+  +===+


          +===+  +===+  +===+  +===+
          |C_1|  |C_2|  |C_3|  |C_4|
          +===+  +===+  +===+  +===+
```
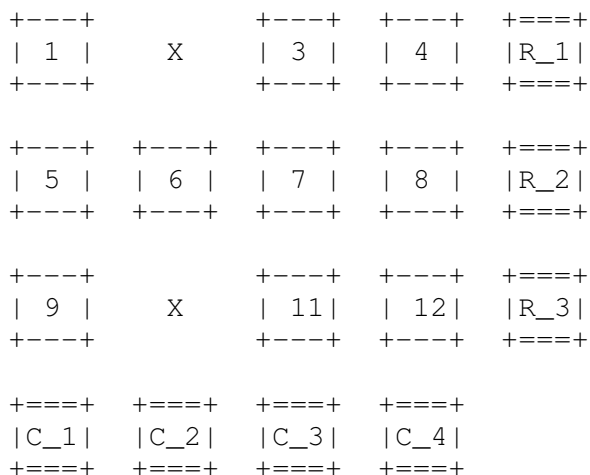
          Figure 17: The resulting pattern after the first iteration

   Since the if condition holds true, the receiver runs a new iteration.
   In the second iteration, source packets #2 and #10 are recovered,
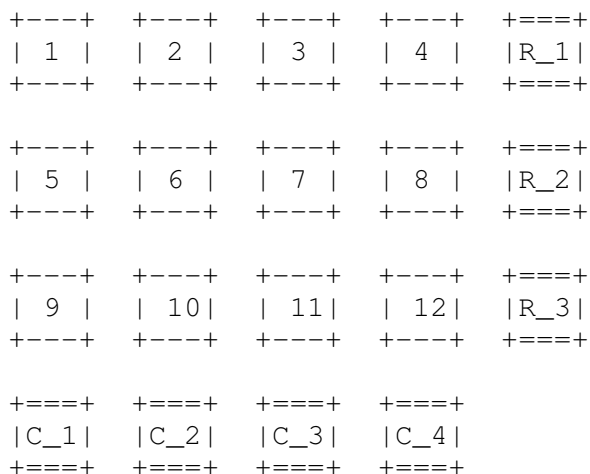   resulting in a full recovery as sketched in Figure 18.


```
          +---+  +---+  +---+  +---+  +===+
          | 1 |  | 2 |  | 3 |  | 4 |  |R_1|
          +---+  +---+  +---+  +---+  +===+


          +---+  +---+  +---+  +---+  +===+
          | 5 |  | 6 |  | 7 |  | 8 |  |R_2|
          +---+  +---+  +---+  +---+  +===+


          +---+  +---+  +---+  +---+  +===+
          | 9 |  | 10|  | 11|  | 12|  |R_3|
          +---+  +---+  +---+  +---+  +===+


          +===+  +===+  +===+  +===+
          |C_1|  |C_2|  |C_3|  |C_4|
          +===+  +===+  +===+  +===+
```

          Figure 18: The resulting pattern after the second iteration

7. SDP Examples

   This section provides two SDP [RFC4566] examples. The examples use
   the FEC grouping semantics defined in [RFC5956].

7.1. Example SDP for Flexible FEC Protection with in-band SSRC mapping

   In this example, we have one source video stream and one FEC repair
   stream. The source and repair streams are multiplexed on different
   SSRCs. The repair window is set to 200 ms.

```
     v=0
     o=mo 1122334455 1122334466 IN IP4 fec.example.com
     s=FlexFEC minimal SDP signalling Example
     t=0 0
     m=video 30000 RTP/AVP 96 98
     c=IN IP4 143.163.151.157
     a=rtpmap:96 VP8/90000
     a=rtpmap:98 flexfec/90000
     a=fmtp:98; repair-window=200ms
```

7.2. Example SDP for Flex FEC Protection with explicit signalling in
      the SDP

   In this example, we have one source video stream (ssrc:1234) and one
   FEC repair streams (ssrc:2345). We form one FEC group with the
   "a=ssrc-group:FEC-FR 1234 2345" line. The source and repair streams
   are multiplexed on different SSRCs. The repair window is set to 200
   ms.

```
     v=0
     o=ali 1122334455 1122334466 IN IP4 fec.example.com
     s=2-D Parity FEC with no in band signalling Example
     t=0 0
     m=video 30000 RTP/AVP 100 110
     c=IN IP4 233.252.0.1/127
     a=rtpmap:100 MP2T/90000
     a=rtpmap:110 flexfec/90000
     a=fmtp:110 L:5; D:10; ToP:2; repair-window:200000
     a=ssrc:1234
     a=ssrc:2345
     a=ssrc-group:FEC-FR 1234 2345
```

8. Congestion Control Considerations

   FEC is an effective approach to provide applications resiliency
   against packet losses. However, in networks where the congestion is
   a major contributor to the packet loss, the potential impacts of
   using FEC SHOULD be considered carefully before injecting the repair

flows into the network. In particular, in bandwidth-limited
networks, FEC repair flows may consume most or all of the available
bandwidth and consequently may congest the network. In such cases,
the applications MUST NOT arbitrarily increase the amount of FEC
protection since doing so may lead to a congestion collapse. If
desired, stronger FEC protection MAY be applied only after the source
rate has been reduced [I-D.singh-rmcat-adaptive-fec].

In a network-friendly implementation, an application SHOULD NOT send/
receive FEC repair flows if it knows that sending/receiving those FEC
repair flows would not help at all in recovering the missing packets.
However, it MAY still continue to use FEC if considered for bandwidth
estimation instead of speculatively probe for additional capacity
[Holmer13][Nagy14]. It is RECOMMENDED that the amount of FEC
protection is adjusted dynamically based on the packet loss rate
observed by the applications.

In multicast scenarios, it may be difficult to optimize the FEC
protection per receiver. If there is a large variation among the
levels of FEC protection needed by different receivers, it is
RECOMMENDED that the sender offers multiple repair flows with
different levels of FEC protection and the receivers join the
corresponding multicast sessions to receive the repair flow(s) that
is best for them.

Editor's note: Additional congestion control considerations regarding
the use of 2-D parity codes should be added here.

9. Security Considerations

RTP packets using the payload format defined in this specification
are subject to the security considerations discussed in the RTP
specification [RFC3550] 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 encrypting 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, transport and signaling protocol employed. Therefore, a
single mechanism is not sufficient, although if suitable, using the

Secure Real-time Transport Protocol (SRTP) [RFC3711] is recommended. Other mechanisms that may be used are IPsec [RFC4301] and Transport Layer Security (TLS) [RFC5246] (RTP over TCP); other alternatives may exist.

10. IANA Considerations

New media subtypes are subject to IANA registration. For the registration of the payload formats and their parameters introduced in this document, refer to Section 5.

11. Acknowledgments

Some parts of this document are borrowed from [RFC5109]. Thus, the author would like to thank the editor of [RFC5109] and those who contributed to [RFC5109].

Thanks to Bernard Aboba , Rasmus Brandt , Roni Even , Stefan Holmer , Jonathan Lennox , and Magnus Westerlund for providing valuable feedback on earlier versions of this draft.

12. Change Log

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

12.1. draft-ietf-payload-flexible-fec-scheme-03

FEC packet format changed as per discussions in IETF96, Berlin.

Removed section on non-parity codes and flexfec-raptor.

12.2. draft-ietf-payload-flexible-fec-scheme-02

FEC packet format changed as per discussions in IETF94, Tokyo.

Added section on non-parity codes.

Registration of application/flexfec-raptor.

12.3. draft-ietf-payload-flexible-fec-scheme-01

FEC packet format changed as per discussions in IETF93, Prague.

Replaced non-interleaved-parityfec and interleaved-parity-fec with flexfec.

SDP simplified for the case when association to RTP is made in the
FEC header and not in the SDP.

12.4. draft-ietf-payload-flexible-fec-scheme-00

Initial WG version, based on draft-singh-payload-1d2d-parity-scheme-
00.

12.5. draft-singh-payload-1d2d-parity-scheme-00

This is the initial version, which is based on draft-ietf-fecframe-
1d2d-parity-scheme-00. The following are the major changes compared
to that document:

o Updated packet format with 16-, 48-, 112- bitmask.

o Updated the sections on: repair packet construction, source packet
   construction.

o Updated the media type registration and aligned to RFC6838.

12.6. draft-ietf-fecframe-1d2d-parity-scheme-00

o Some details were added regarding the use of CNAME field.

o Offer-Answer and Declarative Considerations sections have been
   completed.

o Security Considerations section has been completed.

o The timestamp field definition has changed.

13. References

13.1. 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,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model
              with Session Description Protocol (SDP)", RFC 3264, DOI
              10.17487/RFC3264, June 2002,
              <http://www.rfc-editor.org/info/rfc3264>.

   [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
              Jacobson, "RTP: A Transport Protocol for Real-Time
              Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550,
              July 2003, <http://www.rfc-editor.org/info/rfc3550>.

   [RFC3555] Casner, S. and P. Hoschka, "MIME Type Registration of RTP
              Payload Formats", RFC 3555, DOI 10.17487/RFC3555, July
              2003, <http://www.rfc-editor.org/info/rfc3555>.

   [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
              Description Protocol", RFC 4566, DOI 10.17487/RFC4566,
              July 2006, <http://www.rfc-editor.org/info/rfc4566>.

   [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in
              the Session Description Protocol", RFC 5956, DOI 10.17487/
              RFC5956, September 2010,
              <http://www.rfc-editor.org/info/rfc5956>.

   [RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error
              Correction (FEC) Framework", RFC 6363, DOI 10.17487/
              RFC6363, October 2011,
              <http://www.rfc-editor.org/info/rfc6363>.

   [RFC6709] Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design
              Considerations for Protocol Extensions", RFC 6709, DOI
              10.17487/RFC6709, September 2012,
              <http://www.rfc-editor.org/info/rfc6709>.

   [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type
              Specifications and Registration Procedures", BCP 13, RFC
              6838, DOI 10.17487/RFC6838, January 2013,
              <http://www.rfc-editor.org/info/rfc6838>.

   [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla,
              "Guidelines for Choosing RTP Control Protocol (RTCP)
              Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/RFC7022,
              September 2013, <http://www.rfc-editor.org/info/rfc7022>.

13.2. Informative References

   [Holmer13]
              Holmer, S., Shemer, M., and M. Paniconi, "Handling Packet
              Loss in WebRTC", Proc. of IEEE International Conference on
              Image Processing (ICIP 2013) , 9 2013.

   [I-D.singh-rmcat-adaptive-fec]
              Varun, V., Nagy, M., Ott, J., and L. Eggert, "Congestion
              Control Using FEC for Conversational Media", draft-singh-
              rmcat-adaptive-fec-03 (work in progress), March 2016.

   [Nagy14]   Nagy, M., Singh, V., Ott, J., and L. Eggert, "Congestion
              Control using FEC for Conversational Multimedia
              Communication", Proc. of 5th ACM Internation Conference on
              Multimedia Systems (MMSys 2014) , 3 2014.

   [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time
              Streaming Protocol (RTSP)", RFC 2326, DOI 10.17487/
              RFC2326, April 1998,
              <http://www.rfc-editor.org/info/rfc2326>.

   [RFC2733] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format
              for Generic Forward Error Correction", RFC 2733, DOI
              10.17487/RFC2733, December 1999,
              <http://www.rfc-editor.org/info/rfc2733>.

   [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session
              Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974,
              October 2000, <http://www.rfc-editor.org/info/rfc2974>.

   [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
              Norrman, "The Secure Real-time Transport Protocol (SRTP)",
              RFC 3711, DOI 10.17487/RFC3711, March 2004,
              <http://www.rfc-editor.org/info/rfc3711>.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, DOI 10.17487/RFC4301,
              December 2005, <http://www.rfc-editor.org/info/rfc4301>.

   [RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error
              Correction", RFC 5109, DOI 10.17487/RFC5109, December
              2007, <http://www.rfc-editor.org/info/rfc5109>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/
              RFC5246, August 2008,
              <http://www.rfc-editor.org/info/rfc5246>.

   [SMPTE2022-1]
              SMPTE 2022-1-2007, , "Forward Error Correction for Real-
              Time Video/Audio Transport over IP Networks", 2007.

Authors' Addresses

   Varun Singh
   CALLSTATS I/O Oy
   Runeberginkatu 4c A 4
   Helsinki 00100
   Finland

   Email: varun.singh@iki.fi
   URI: http://www.callstats.io/

   Ali Begen
   Networked Media
   Konya
   Turkey

   Email: ali.begen@networked.media

   Mo Zanaty
   Cisco
   Raleigh, NC
   USA

   Email: mzanaty@cisco.com

   Giridhar Mandyam
   Qualcomm Innovation Center
   5775 Morehouse Drive
   San Diego, CA 92121
   USA

   Phone: +1 858 651 7200
   Email: mandyam@qti.qualcomm.com