Network Working Group
Request for Comments: 5109
Obsoletes: 2733, 3009
Category: Standards Track

A. Li, Ed.

December 2007

RTP Payload Format for Generic Forward Error Correction

Status of This Memo

Abstract

This document specifies a payload format for generic Forward Error Correction (FEC) for media data encapsulated in RTP. It is based on the exclusive-or (parity) operation. The payload format described in this document allows end systems to apply protection using various protection lengths and levels, in addition to using various protection group sizes to adapt to different media and channel characteristics. It enables complete recovery of the protected packets or partial recovery of the critical parts of the payload depending on the packet loss situation. This scheme is completely compatible with non-FEC-capable hosts, so the receivers in a multicast group that do not implement FEC can still work by simply ignoring the protection data. This specification obsoletes RFC 2733 and RFC 3009. The FEC specified in this document is not backward compatible with RFC 2733 and RFC 3009.

Table of Contents

1. Introduction

The nature of real-time applications implies that they usually have more stringent delay requirements than normal data transmissions. As a result, retransmission of the lost packets is generally not a valid option for such applications. In these cases, a better method to attempt recovery of information from packet loss is through Forward Error Correction (FEC). FEC is one of the main methods used to

protect against packet loss over packet-switched networks [9, 10]. In particular, the use of traditional error correcting codes, such as parity, Reed-Solomon, and Hamming codes, has seen much application. To apply these mechanisms, protocol support is required. RFC 2733 [9] and RFC 3009 [11] defined one of such FEC protocols. However, in these two RFCs a few fields (the P, X, and CC fields) in the RTP header are specified in ways that are not consistent as they are designed in RTP [1]. This prevents the payload-independent validity check of the RTP packets.

This document extends the FEC defined in RFC 2733 and RFC 3009 to include unequal error protection on the payload data. It specifies a general algorithm with the two previous RFCs as its special cases. This specification also fixes the above-mentioned inconsistency with RFC 2733 and RFC 3009, and will obsolete those two previous RFCs. Please note that the payload specified in this document is not backward compatible with RFC 2733 and RFC 3009. Because the payload specified in this document is signaled by different MIMEs from those of RFC 3009, there is no concern of misidentification of different parity FEC versions in capacity exchange. For parity FECs specified here and in RFC 2733 and RFC 3009, the payload data are unaltered and additional FEC data are sent along to protect the payload data. Hence, the communication of the payload data would flow without problem between hosts of different parity FEC versions and hosts that did not implement parity FEC. The receiving hosts with incompatible FEC from the sending host would not be able to benefit from the additional FEC data, so it is recommended that existing host implementing RFC 2733 and RFC 3009 should be updated to follow this specification when possible.

This document defines a payload format for RTP [1] that allows for generic forward error correction of real-time media. In this context, generic means that the FEC protocol is (1) independent of the nature of the media being protected, be it audio, video, or otherwise; (2) flexible enough to support a wide variety of FEC configurations; (3) designed for adaptivity so that the FEC technique can be modified easily without out-of-band signaling; and (4) supportive of a number of different mechanisms for transporting the FEC packets.

Furthermore, in many scenarios the bandwidth of the network connections is a very limited resource. On the other hand, most of the traditional FEC schemes are not designed for optimal utilization of the limited bandwidth resource. An often used improvement is unequal error protection that provides different levels of protection for different parts of the data stream, which vary in importance. The unequal error protection schemes can usually make more efficient use of bandwidth to provide better overall protection of the data

stream against the loss. Proper protocol support is essential for realizing these unequal error protection mechanisms. The application of most of the unequal error protection schemes requires having the knowledge of the importance for different parts of the data stream. For that reason, most of such schemes are designed for particular types of media according to the structure of the media protected, and as a result, are not generic.

The FEC algorithm and protocol are defined in this document for generic forward error correction with unequal error protection for real-time media. The particular algorithm defined here is called the Uneven Level Protection (ULP). The payload data are protected by one or more protection levels. Lower protection levels can provide greater protection by using smaller group sizes (compared to higher protection levels) for generating the FEC packet. As we will discuss below, audio/video applications would generally benefit from unequal error protection schemes that give more protection to the beginning part of each packet such as ULP. The data that are closer to the beginning of the packet are in general more important and tend to carry more information than the data farther behind in the packet.

It is well known that in many multimedia streams the more important parts of the data are always at the beginning of the data packet. This is the common practice in codec design since the beginning of the packet is closer to the re-synchronization marker at the header and thus is more likely to be correctly decoded. In addition, almost all media formats have the frame headers at the beginning of the packet, which is the most vital part of the packet.

For video streams, most modern formats have optional data partitioning modes to improve error resilience in which the video macroblock header data, motion vector data, and Discrete Cosine Transform (DCT) coefficient data are separated into their individual partitions. For example, in ITU-T H.263 version 3, there is the optional data partitioned syntax of Annex V. In MPEG-4 Visual Simple Profile, there is the optional data partitioning mode. When these modes are enabled, the video macroblock (MB) header and motion vector partitions (which are much more important to the quality of the video reconstruction) are transmitted in the partition(s) at the beginning of the video packet while residue DCT coefficient partitions (which are less important) are transmitted in the partition close to the end of the packet. Because the data is arranged in descending order of importance, it would be beneficial to provide more protection to the beginning part of the packet in transmission.

For audio streams, the bitstreams generated by many of the new audio codecs also contain data with different classes of importance. These different classes are then transmitted in order of descending

importance. Applying more protection to the beginning of the packet would also be beneficial in these cases. Even for uniform-significance audio streams, various time shifting and stretching techniques can be applied to the partially recovered audio data packets.

Audio/video applications would generally benefit from the FEC algorithms specified in this document. With ULP, the efficiency of the protection of the media payload can potentially be further improved. This document specifies the protocol and algorithm for applying the generic FEC to the RTP media payloads.

2. Terminology

The following terms are used throughout this document:

Media Payload: The raw, unprotected user data that are transmitted from the sender. The media payload is placed inside of an RTP packet.

Media Header: The RTP header for the packet containing the media payload.

Media Packet: The combination of a media payload and media header is called a media packet.

FEC Packet: The FEC algorithms at the transmitter take the media packets as an input. They output both the media packets that they are passed, and newly generated packets called FEC packets, which contain redundant media data used for error correction. The FEC packets are formatted according to the rules specified in this document.

FEC Header: The header information contained in an FEC packet.

FEC Level Header: The header information contained in an FEC packet for each level.

FEC Payload: The payload of an FEC packet. It may be divided into multiple levels.

Associated: A FEC packet is said to be "associated" with one or more media packets (or vice versa) when those media packets are used to generate the FEC packet (by use of the exclusive-or operation). It refers to only those packets used to generate the level 0 FEC payload, if not explicitly stated otherwise.

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 [2].

3. Basic Operation

The payload format described here is used when the sender in an RTP session would like to protect the media stream it is sending with generic parity FEC. The FEC supported by this format is based on simple exclusive-or (XOR) parities operation. The sender takes the packets from the media stream requiring protection and determines the protection levels for these packets and the protection length for each level. The data are grouped together as described below in Section 7. The XOR operation is applied across the payload to generate the FEC information. The results following the procedures defined here are RTP packets containing FEC information. These packets can be used at the receiver to recover the packets or parts of the packets used to generate the FEC information.

The payload format for FEC contains information that allows the sender to tell the receiver exactly which media packets are protected by the FEC packet, and the protection levels and lengths for each of the levels. Specifically, each FEC packet contains an offset mask m(k) for each protection level k. If the bit i in the mask m(k) is set to 1, then media packet number N + i is protected by this FEC packet at level k. N is called the sequence number base, and is sent in the FEC packet as well. The amount of data that is protected at level k is indicated by L(k), which is also sent in the FEC packet. The protection length, offset mask, payload type, and sequence number base fully identify the parity code applied to generate the FEC packet with little overhead. A set of rules is described in Section 7.4 that defines how the mask should be set for different protection levels, with examples in Section 10.

This document also describes procedures on transmitting all the protection operation parameters in-band. This allows the sender great flexibility; the sender can adapt the protection to current network conditions and be certain the receivers can still make use of the FEC for recovery.

At the receiver, both the FEC and original media are received. If no media packets are lost, the FEC packets can be ignored. In the event of a loss, the FEC packets can be combined with other received media to recover all or part of the missing media packets.

4. Parity Codes

For brevity, we define the function f(x,y,..) to be the XOR (parity) operator applied to the data blocks x,y,... The output of this function is another block, called the parity block. For simplicity, we assume here that the parity block is computed as the bitwise XOR of the input blocks. The exact procedure is specified in Section 8.

Protection of data blocks using parity codes is accomplished by generating one or more parity blocks over a group of data blocks. To be most effective, the parity blocks must be generated by linearly independent combinations of data blocks. The particular combination is called a parity code. The payload format uses XOR parity codes.

For example, consider a parity code that generates a single parity block over two data blocks. If the original media packets are a,b,c,d, the packets generated by the sender are:

```
   a        b        c        d              <-- media stream
              f(a,b)            f(c,d)       <-- FEC stream
```

where time increases to the right. In this example, the error correction scheme (we use the terms scheme and code interchangeably) introduces a 50% overhead. But if b is lost, a and f(a,b) can be used to recover b.

It may be useful to point out that there are many other types of forward error correction codes that can also be used to protect the payload besides the XOR parity code. One notable example is Reed-Solomon code, and there are many others [12]. However, XOR parity code is used here because of its effectiveness and simplicity in both protocol design and implementation. This is particularly important for implementation in nodes with limited resources.

5. Uneven Level Protection (ULP)

As we can see from the simple example above, the protection on the data depends on the size of the group. In the above example, the group size is 2. So if any one of the three packets (two payload packets and one FEC packet) is lost, the original payload data can still be recovered.

In general, the FEC protection operation is a trade-off between the bandwidth and the protection strength. The more FEC packets that are generated as a fraction of the source media packets, the stronger the protection against loss but the greater the bandwidth consumed by the combined stream.

As is the common case in most of the media payload, not all the parts of the packets are of the same importance. Using this property, one can potentially achieve more efficient use of the channel bandwidth using unequal error protection, i.e., applying different protection for different parts of the packet. More bandwidth is spent on protecting the more important parts, while less bandwidth on the less important parts.

The packets are separated into sections of decreasing importance, and protection of different strength is applied to each portion - the sections are known as "levels". The protection operation is applied independently at each level. A single FEC packet can carry parity data for multiple levels. This algorithm is called uneven level protection, or ULP.

The protection of ULP is illustrated in Figure 1 below. In this example, two ULP FEC packets are protecting four payload packets.

ULP FEC packet #1 has only one level, which protects packets A and B. Instead of applying parity operation to the entire packets of A and B, it only protects a length of data of both packets. The length, which can be chosen and changed dynamically during a session, is called the protection length.

ULP FEC packet #2 has two protection levels. The level 0 protection is the same as for ULP FEC packet #1 except that it is operating on packets C and D. The level 1 protection is using parity operation applied on data from packets A, B, C, and D. Note that level 1 protection operates on a different set of packets from level 0 and has a different protection length from level 0, so are any other levels. Information is all conveyed in-band through the protocols specified in this document.

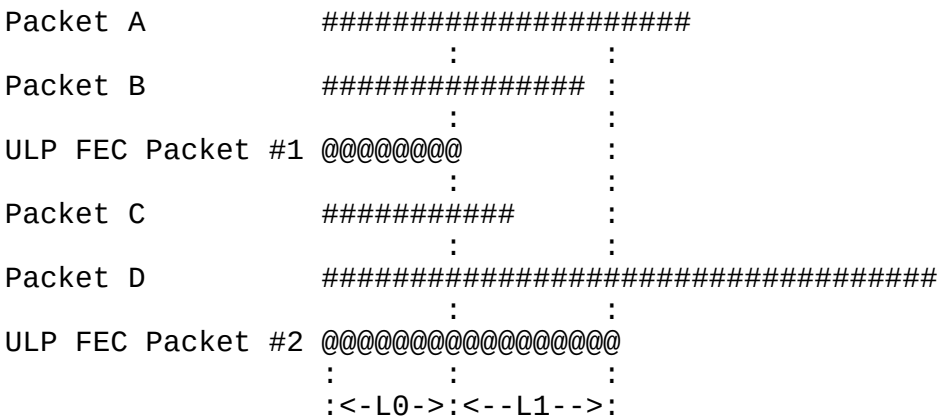


Figure 1: Unequal Level Protection

As we have discussed in the introduction, media streams usually have the more important parts at the beginning of the packet. It is usually useful to have the stronger protection in the levels closer to the beginning of the packet, and weaker protection in the levels farther back. ULP algorithm provides such FEC protection.

ULP FEC not only provides more protection to the beginning of the packet (which is more important), it also avoids as much as possible the less efficient scenarios that an earlier section of a packet is unrecoverable while a later section can be recovered (and often has to be discarded).

6. RTP Media Packet Structure

The formatting of the media packets is unaffected by FEC. If the FEC is sent as a separate stream, the media packets are sent as if there was no FEC.

This approach has the advantage that media packets can be interpreted by receivers that do not support FEC. This compatibility with non-FEC capable receivers is particularly useful in the multicast scenarios. The overhead for using the FEC scheme is only present in FEC packets, and can be easily monitored and adjusted by tracking the amount of FEC in use.

7. FEC Packet Structure

7.1. Packet Structure

A FEC packet is constructed by placing an FEC header and one or more levels of FEC header and payload into the RTP payload, as shown in Figure 2:

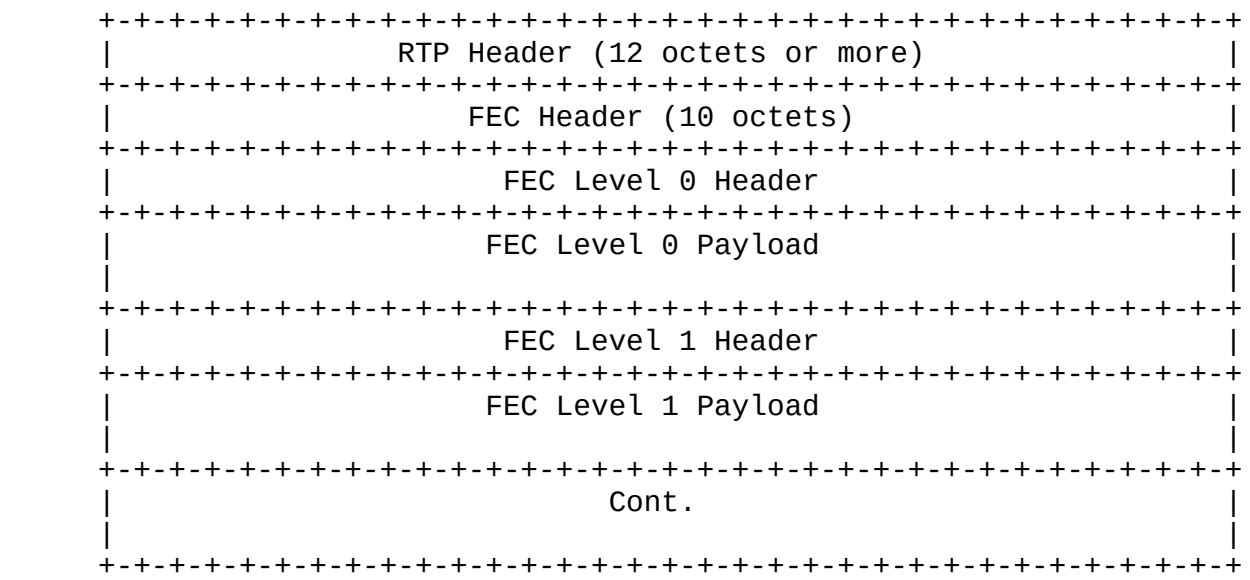


Figure 2: FEC Packet Structure

7.2. RTP Header for FEC Packets

The RTP header for FEC packets is only used when the FEC are sent in a separate stream from the protected payload stream (as defined in Section 14). Hence, much of the discussion below applies only to that scenario. All the fields in the RTP header of FEC packets are used according to RFC 3550 [1], with some of them further clarified below.

Marker: This field is not used for this payload type, and SHALL be set to 0.

Synchronization Source (SSRC): The SSRC value SHALL be the same as the SSRC value of the media stream it protects.

Sequence Number (SN): The sequence number has the standard definition - it MUST be one higher than the sequence number in the previously transmitted FEC packet.

Timestamp (TS): The timestamp MUST be set to the value of the media RTP clock at the instant the FEC packet is transmitted. Thus, the TS value in FEC packets is always monotonically increasing.

Payload type: The payload type for the FEC packets is determined through dynamic, out-of-band means. According to RFC 3550 [1], RTP participants that cannot recognize a payload type must discard it. This provides backward compatibility. The FEC mechanisms can then be

used in a multicast group with mixed FEC-capable and FEC-incapable receivers, particularly when the FEC protection is sent as redundant encoding (see Section 14). In such cases, the FEC protection will have a payload type that is not recognized by the FEC-incapable receivers, and will thus be disregarded.

7.3. FEC Header for FEC Packets

The FEC header is 10 octets. The format of the header is shown in Figure 3 and consists of extension flag (E bit), long-mask flag (L bit), P recovery field, X recovery field, CC recovery field, M recovery field, PT recovery field, SN base field, TS recovery field, and length recovery field.

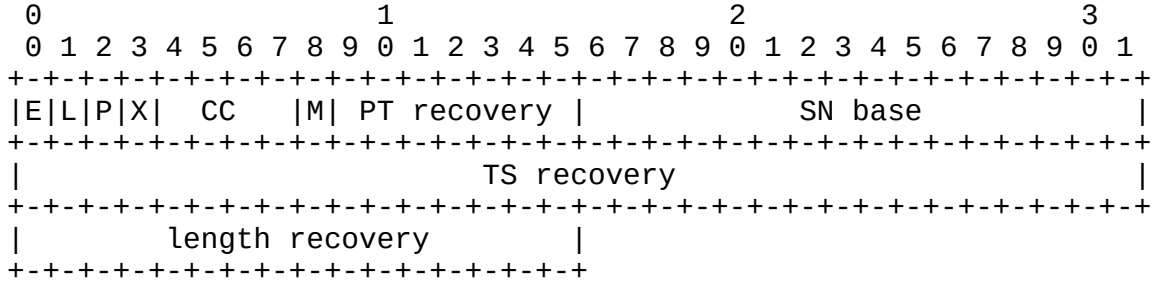


Figure 3: FEC Header Format

The E bit is the extension flag reserved to indicate any future extension to this specification. It SHALL be set to 0, and SHOULD be ignored by the receiver.

The L bit indicates whether the long mask is used. When the L bit is not set, the mask is 16 bits long. When the L bit is set, the mask is then 48 bits long.

The P recovery field, the X recovery field, the CC recovery field, the M recovery field, and the PT recovery field are obtained via the protection operation applied to the corresponding P, X, CC, M, and PT values from the RTP header of the media packets associated with the FEC packet.

The SN base field MUST be set to the lowest sequence number, taking wrap around into account, of those media packets protected by FEC (at all levels). This allows for the FEC operation to extend over any string of at most 16 packets when the L field is set to 0, or 48 packets when the L field is set to 1, and so on.

The TS recovery field is computed via the protection operation applied to the timestamps of the media packets associated with this FEC packet. This allows the timestamp to be completely recovered.

The length recovery field is used to determine the length of any recovered packets. It is computed via the protection operation applied to the unsigned network-ordered 16-bit representation of the sums of the lengths (in bytes) of the media payload, CSRC list, extension and padding of each of the media packets associated with this FEC packet (in other words, the CSRC list, RTP extension, and padding of the media payload packets, if present, are "counted" as part of the payload). This allows the FEC procedure to be applied even when the lengths of the protected media packets are not identical. For example, assume that an FEC packet is being generated by xor'ing two media packets together. The length of the payload of two media packets is 3 (0b011) and 5 (0b101) bytes, respectively. The length recovery field is then encoded as 0b011 xor 0b101 = 0b110.

7.4. FEC Level Header for FEC Packets

The FEC level header is 4 or 8 octets (depending on the L bit in the FEC header). The formats of the headers are shown in Figure 4.

The FEC level headers consist of a protection length field and a mask field. The protection length field is 16 bits long. The mask field is 16 bits long (when the L bit is not set) or 48 bits long (when the L bit is set).

The mask field in the FEC level header indicates which packets are associated with the FEC packet at the current level. It is either 16 or 48 bits depending on the value of the L bit. If bit i in the mask is set to 1, then the media packet with sequence number N + i is associated with this FEC packet, where N is the SN Base field in the FEC packet header. The most significant bit of the mask corresponds to i=0, and the least significant to i=15 when the L bit is set to 0, or i=47 when the L bit is set to 1.

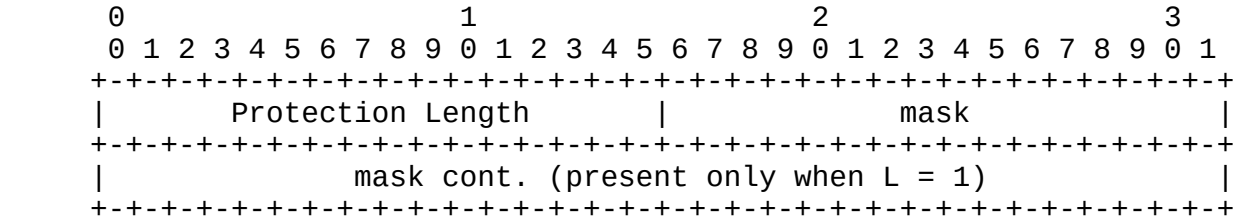


Figure 4: ULP Level Header Format

The setting of the mask field shall follow the following rules:

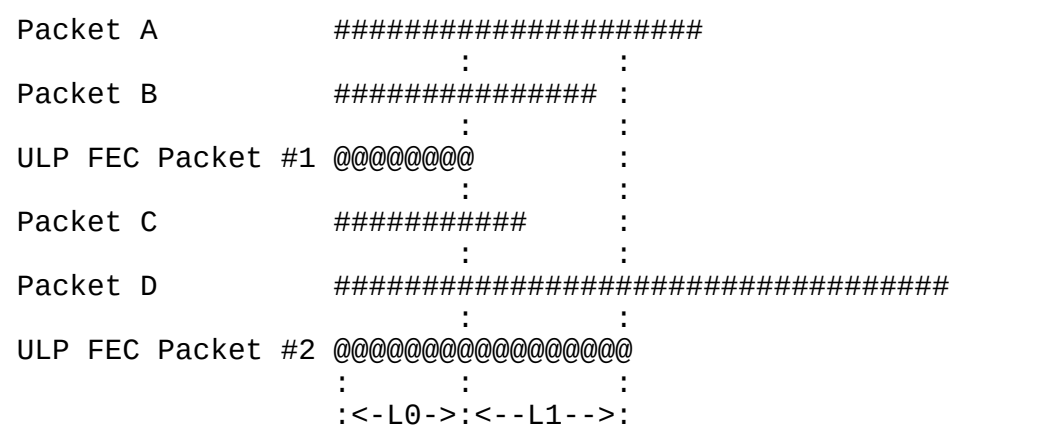
a. A media packet SHALL be protected only once at each protection level higher than level 0. A media packet MAY be protected more than once at level 0 by different packets, providing the protection lengths of level 0 of these packets are equal.

b. For a media packet to be protected at level p, it MUST also be protected at level p-1 in any FEC packets. Please note that the protection level p for a media packet can be in an FEC packet that is different from the one that contains protection level p-1 for the same media packet.

c. If a ULP FEC packet contains protection at level p, it MUST also contain protection at level p-1. Note that the combination of payload packets that are protected in level p may be different from those of level p-1.

The rationale for rule (a) is that multiple protection increases the complexity of the recovery implementation. At higher levels, the multiple protection offers diminishing benefit, so its application is restricted to level 0 for simpler implementation. The rationale for rule (b) is that the protection offset (for each associated packet) is not explicitly signaled in the protocol. With this restriction, the offset can be easily deducted from protection lengths of the levels. The rationale of rule (c) is that the level of protection is not explicitly indicated. This rule is set to implicitly specify the levels.

One example of the protection combinations is illustrated in Figure 5 below. It is the same example as shown in Figure 1. This same example is also shown in more detail in Section 10.2 to illustrate how the fields in the headers are set.



Payload packet #	ULP FEC packet that protects at level	
	L0	L1
A	#1	#2
B	#1	#2
C	#2	#2
D	#2	#2

Figure 5: An Example of Protection Combination

In this example, ULP FEC packet #1 only has protection level 0. ULP FEC packet #2 has protection levels 0 and 1. Read across the table, it is shown that payload packet A is protected by ULP FEC packet #1 at level 0, by ULP FEC packet #2 at level 1, and so on. Also, it can be easily seen from the table that ULP FEC packet #2 protects at level 0 payload packets C and D, at level 1 payload packets A-D, and so on. For additional examples with more details, please refer to Section 10, "Examples".

The payload of the ULP FEC packets of each level is the protection operation (XOR) applied to the media payload and padding of the media packets associated with the ULP FEC packet at that level. Details are described in Section 8 on the protection operation.

The size of the ULP FEC packets is determined by the protection lengths chosen for the protection operation. In the above example, ULP FEC packet #1 has length L0 (plus the header overhead). ULP FEC packet #2 with two levels has length L0+L1 (plus the header overhead). It is longer than some of the packets it protects (packets B and C in this example), and is shorter than some of the packets it protects (packets A and D in this example).

Note that it's possible for the FEC packet (non-ULP and ULP) to be larger than the longest media packets it protects because of the overhead from the headers and/or if a large protection length is chosen for ULP. This could cause difficulties if this results in the FEC packet exceeding the Maximum Transmission Unit size for the path along which it is sent.

8. Protection Operation

FEC packets are formed from an "FEC bit string" that is generated from the data of the protected media RTP packets. More specifically, the FEC bit string is the bitwise exclusive OR of the "protected bit strings" of the protected media RTP packets.

The following procedure MAY be followed for the protection operation. Other procedures MAY be used, but the end result MUST be identical to the one described here.

8.1. Generation of the FEC Header

In the case of the FEC header, the protected bit strings (80 bits in length) are generated for each media packet to be protected at FEC level 0. It is formed by concatenating the following fields together in the order specified:

- The first 64 bits of the RTP header (64 bits)
- Unsigned network-ordered 16-bit representation of the media 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)

After the FEC bit string is formed by applying parity operation on the protected bit strings, the FEC header is generated from the FEC bit string as follows:

The first (most significant) 2 bits in the FEC bit string are skipped. The next bit in the FEC bit string is written into the P recovery bit of the FEC header in the FEC packet. The next bit in the FEC bit string is written into the E recovery bit of the FEC header. The next 4 bits of the FEC bit string are written into the CC recovery field of the FEC header. The next bit is written into the M recovery bit of the FEC header. The next 7 bits of the FEC bit string are written into the PT recovery field in the FEC header. The next 16 bits are skipped. The next 32 bits of the FEC bit string are written into the TS recovery field in the FEC header. The next 16 bits are written into the length recovery field in the packet header.

8.2. Generation of the FEC Payload

For generation of the FEC payload, the protected bit strings are simply the protected RTP packets. The FEC bit string is thus the bitwise exclusive OR of these protected media RTP packets. Such FEC bit strings need to be generated for each level, as the group of protected payload packets may be different for each level. If the lengths of the protected RTP packets are not equal, each shorter packet MUST be padded to the length of the longest packet by adding octet 0 at the end.

For protection level n (n = 0, 1, ...), only Ln octets of data are set as the FEC level n payload data after the level n ULP header. The data is the Ln octets of data starting with the (Sn + 13)th octet in the FEC bit string, where:

Sn = sum(Li : 0 <= i < n).

Li is the protection length of level i, and S0 is defined to be 0. The reason for omitting the first 12 octets is that that information is protected by the FEC header already.

9. Recovery Procedures

The FEC packets allow end systems to recover from the loss of media packets. This section describes the procedure for performing this recovery.

Recovery requires two distinct operations. The first determines which packets (media and FEC) must be combined in order to recover a missing packet. Once this is done, the second step is to actually reconstruct the data. The second step MUST be performed as described below. The first step MAY be based on any algorithm chosen by the implementer. Different algorithms result in a trade-off between complexity and the ability to recover missing packets, if possible.

The lost payload packets may be recovered in full or in parts depending on the data-loss situation due to the nature of unequal error protection (when it is used). The partial recovery of the packet can be detected by checking the recovery length of the packet retrieved from the FEC header against the actual length of the recovered payload data.

9.1. Reconstruction of the RTP Header

Let T be the list of packets (FEC and media) that can be combined to recover some media packet xi at level 0. The procedure is as follows:

1. For the media packets in T, compute the first 80 bits of the protected bit string following the procedure as described for generating the FEC header in the previous section.

2. For the FEC packet in T, the FEC bit string is the 80-bit FEC header.

3. Calculate the recovery bit string as the bitwise exclusive OR of the protected bit string generated from all the media packets in T and the FEC bit string generated from all the FEC packets 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 recovery bit string.

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

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

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

9. Set the marker bit in the new packet to the next bit in the recovery bit string.

10. Set the payload type in the new packet to the next 7 bits in the recovery bit string.

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

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

13. Take the next 16 bits of the recovery bit string. Whatever unsigned integer this represents (assuming network-order), take that many bytes from the recovery bit string and append them to the new packet. This represents the CSRC list, extension, payload, and the padding of the RTP payload.

14. Set the SSRC of the new packet to the SSRC of the media stream it’s protecting, i.e., the SSRC of the media stream to which the FEC stream is associated.

This procedure will recover the header of an RTP packet up to the SSRC field.

9.2. Reconstruction of the RTP Payload

Let T be the list of packets (FEC and media) that can be combined to recover some media packet xi at a certain protection level. The procedure is as follows:

1. Assume that we are reconstructing the data for level n, the first step is to get the protection length of level n (Ln) from the ULP header of level n.

2. For the FEC packets in T, the FEC bit string of level n is FEC level n payload, i.e., the Ln octets of data following the ULP header of level n.

3. For the media packets in T, the protected bit string of level n is Ln octets of data starting with the (Sn + 13)th octet of the packet. Sn is the same as defined in Section 8.2. Note that the protection of level 0 starts from the 13th octet of the media packet after the SSRC field. The information of the first 12 octets are protected by the FEC header.

4. If any of the protected bit strings of level n generated from the media packets are shorter than the protection length of the current level, pad them to that length. The padding of octet 0 MUST be added at the end of the bit string.

5. Calculate the recovery bit string as the bitwise exclusive OR of the protected bit string of level n generated from all the media packets in T and the FEC bit string of level n generated from all the FEC packets in T.

6. The recovery bit string of the current protection level as generated above is combined through concatenation with the recovery bit string of all the other levels to form the (fully or partially) recovered media packet. Note that the recovery bit string of each protection level MUST be placed at the correct location in the recovered media packet for that level based on protection length settings.

7. The total length of the recovered media packet is recovered from the recovery operation at protection level 0 of the recovered media packet. This information can be used to check if the complete recovery operation (of all levels) has recovered the packet to its full length.

The data protected at the lower protection level is recoverable in a majority of the cases if the higher-level protected data is recoverable. This procedure (together with the procedure for the lower protection levels) will usually recover both the header and payload of an RTP packet up to the protection length of the current level.

10. Examples

In the first two examples considered below (Sections 10.1 and 10.2), we assume that the FEC streams are sent through a separate RTP session as described in Section 14.1. For these examples, we assume that four media packets are to be sent, A, B, C, and D, from SSRC 2. Their sequence numbers are 8, 9, 10, and 11, respectively, and have timestamps of 3, 5, 7, and 9, respectively. Packets A and C use payload type 11, and packets B and D use payload type 18. Packet A has 200 bytes of payload, packet B 140, packet C 100, and packet D 340. Packets A and C have their marker bit set.

The third example (Section 10.3) is to illustrate when the FEC data is sent as redundant data with the payload packets.

10.1. An Example Offers Similar Protection as RFC 2733

We can protect the four payload packets to their full length in one single level with one FEC packet. This offers similar protection as RFC 2733. The scheme is as shown in Figure 6.

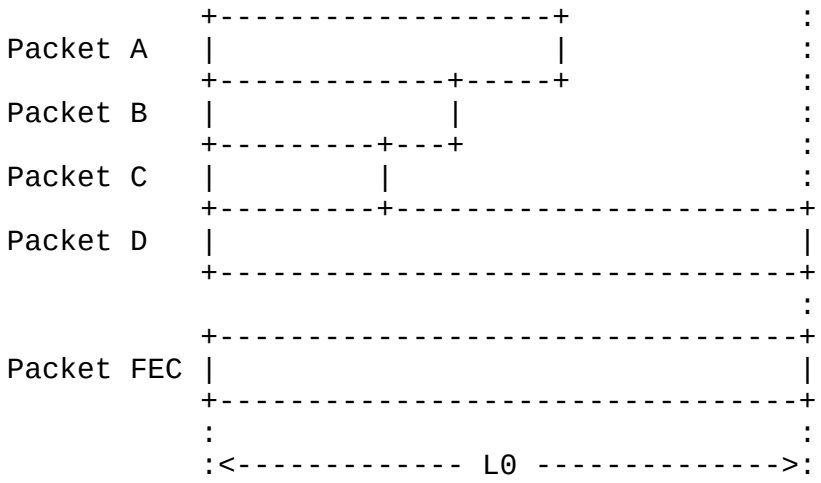
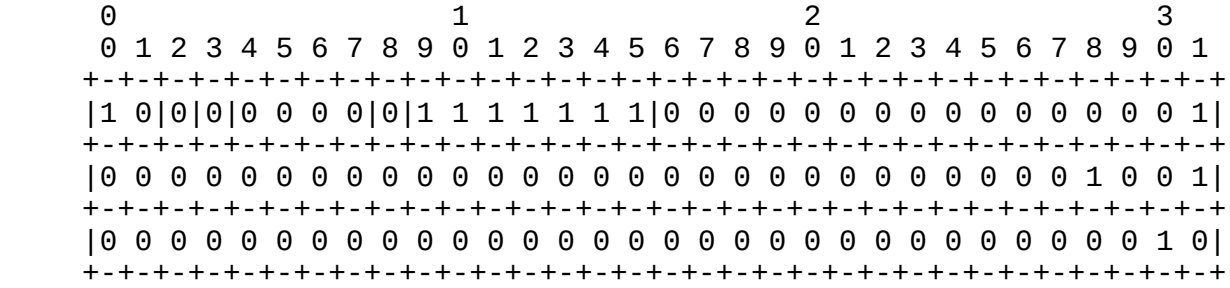


Figure 6: FEC Scheme with Single-Level Protection

An FEC packet is generated from these four packets. We assume that payload type 127 is used to indicate an FEC packet. The resulting RTP header is shown in Figure 7.

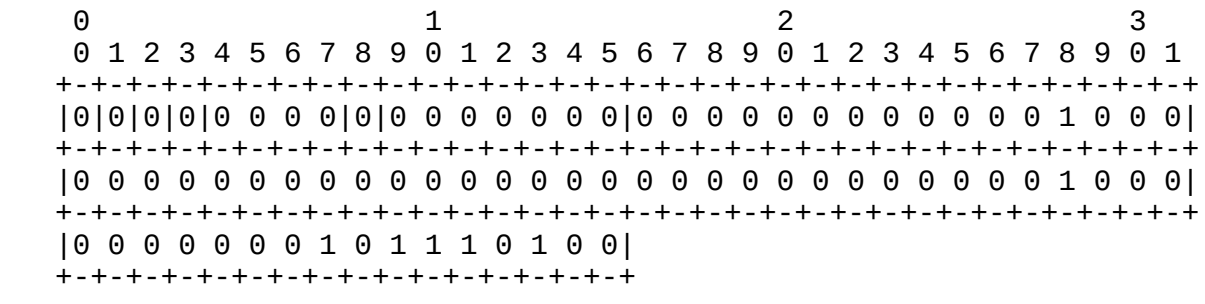
The FEC header in the FEC packet is shown in Figure 8.

The FEC level header for level 0 is shown in Figure 9.



Version: 2
Padding: 0
Extension: 0
Marker: 0
PT: 127
SN: 1
TS: 9
SSRC: 2

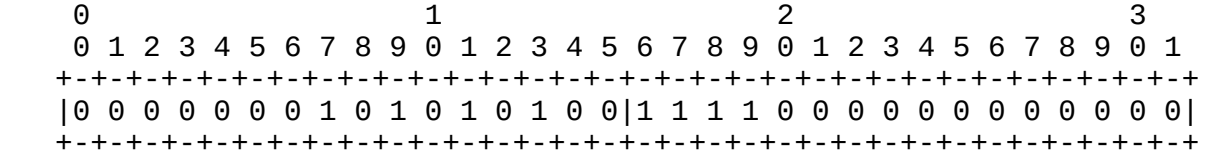
Figure 7: RTP Header of FEC Packet



```
 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|0|0|0 0 0 0|0|0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   E:         0     [this specification]
   L:         0     [short 16-bit mask]
   P rec.:    0     [0 XOR 0 XOR 0 XOR 0]
   X rec.:    0     [0 XOR 0 XOR 0 XOR 0]
   CC rec.:   0     [0 XOR 0 XOR 0 XOR 0]
   M rec.:    0     [1 XOR 0 XOR 1 XOR 0]
   PT rec.:   0     [11 XOR 18 XOR 11 XOR 18]
   SN base:   8     [min(8,9,10,11)]
   TS rec.:   8     [3 XOR 5 XOR 7 XOR 9]
   len. rec.: 372   [200 XOR 140 XOR 100 XOR 340]
```

Figure 8: FEC Header of FEC Packet



```
 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 0 0 0 0 0 1 0 1 0 1 0 1 0 0|1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   L0:        340   [the longest of 200, 140, 100, and 340]
   mask:      61440 [with Bits 1, 2, 3, and 4 marked accordingly for
                     Packets 8, 9, 10, and 11]

   The payload length for level 0 is 340 bytes.
```

Figure 9: FEC Level Header (Level 0)

10.2. An Example with Two Protection Levels

A more complex example is to use FEC at two levels. The level 0 FEC will provide greater protection to the beginning part of the payload packets. The level 1 FEC will apply additional protection to the rest of the packets. This is illustrated in Figure 10. In this example, L0 = 70 and L1 = 90.

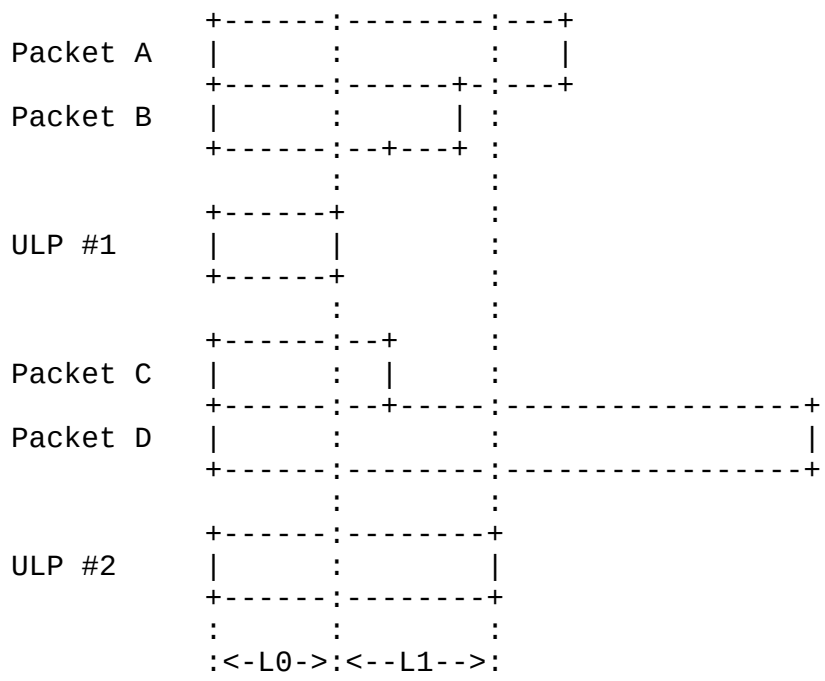
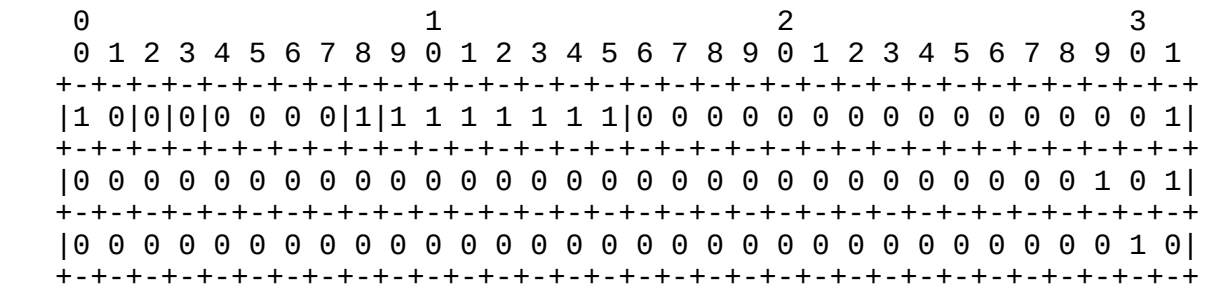


Figure 10: ULP FEC Scheme with Protection Level 0 and Level 1

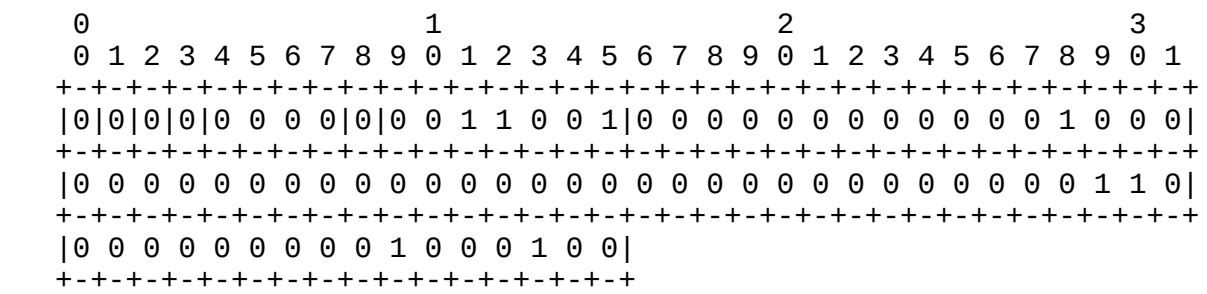
This will result in two FEC packets - #1 and #2.

The resulting ULP FEC packet #1 will have the RTP header as shown in Figure 11. The FEC header for ULP FEC packet #1 will be as shown in Figure 12. The level 0 ULP header for #1 will be as shown in Figure 13.



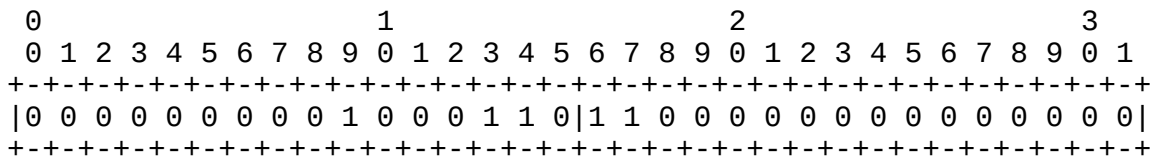
```
   Version:   2
   Padding:   0
   Extension: 0
   Marker:    1
   PT:        127
   SN:        1
   TS:        5
   SSRC:      2
```

Figure 11: RTP Header of FEC Packet #1



```
   E:         0     [this specification]
   L:         0     [short 16-bit mask]
   P rec.:    0     [0 XOR 0 XOR 0 XOR 0]
   X rec.:    0     [0 XOR 0 XOR 0 XOR 0]
   CC rec.:   0     [0 XOR 0 XOR 0 XOR 0]
   M rec.:    0     [1 XOR 0 XOR 1 XOR 0]
   PT rec.:   25    [11 XOR 18]
   SN base:   8     [min(8,9)]
   TS rec.:   6     [3 XOR 5]
   len. rec.: 68    [200 XOR 140]
```

Figure 12: FEC Header of ULP FEC Packet #1

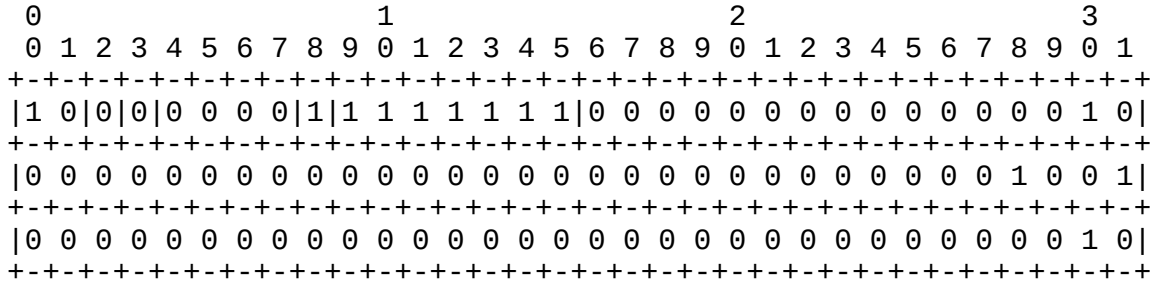


L0: 70
mask: 49152 [with Bits 1 and 2 marked accordingly for Packets 8 and 9]

The payload length for level 0 is 70 bytes.

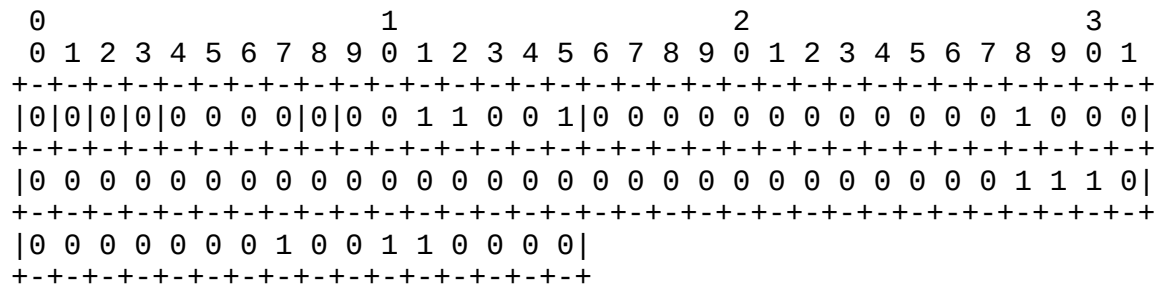
Figure 13: FEC Level Header (Level 0) for FEC Packet #1

The resulting FEC packet #2 will have the RTP header as shown in Figure 14. The FEC header for FEC packet #2 will be as shown in Figure 15. The level 0 ULP header for #2 will be as shown in Figure 16. The level 1 ULP header for #2 will be as shown in Figure 17.



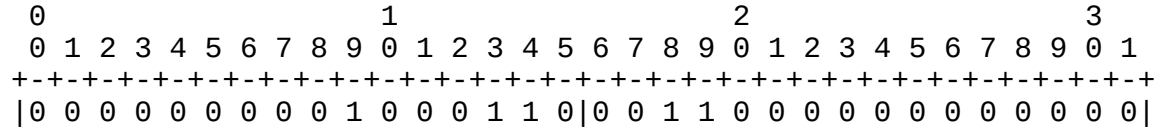
Version: 2
Padding: 0
Extension: 0
Marker: 1
PT: 127
SN: 2
TS: 9
SSRC: 2

Figure 14: RTP Header of FEC Packet #2



E:	0	[this specification]
L:	0	[short 16-bit mask]
P rec.:	0	[0 XOR 0 XOR 0 XOR 0]
X rec.:	0	[0 XOR 0 XOR 0 XOR 0]
CC rec.:	0	[0 XOR 0 XOR 0 XOR 0]
M rec.:	0	[1 XOR 0 XOR 1 XOR 0]
PT rec.:	25	[11 XOR 18]
SN base:	8	[min(8,9,10,11)]
TS rec.:	14	[7 XOR 9]
len. rec.:	304	[100 XOR 340]

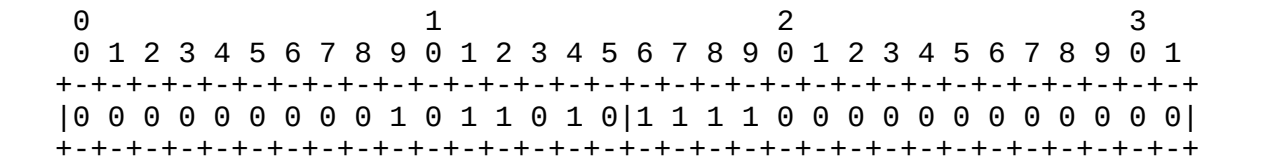
Figure 15: FEC Header of FEC Packet #2



L0:	70	
mask:	12288	[with Bits 3 and 4 marked accordingly for Packets 10 and 11]

The payload length for level 0 is 70 bytes.

Figure 16: FEC Level Header (Level 0) for FEC Packet #2



L1: 90
mask: 61440 [with Bits 1, 2, 3, and 4 marked accordingly for Packets 8, 9, 10, and 11]

The payload length for level 1 is 90 bytes.

Figure 17: FEC Level Header (Level 1) for FEC Packet #2

10.3. An Example with FEC as Redundant Coding

This example illustrates FEC sent as redundant coding in the same stream as the payload. We assume that five media packets are to be sent, A, B, C, D, and E, from SSRC 2. Their sequence numbers are 8, 9, 10, 11, and 12, respectively, and have timestamps of 3, 5, 7, 9, and 11, respectively. All the media data is coded with primary coding (and FEC as redundant coding only protects the primary coding) and uses payload type 11. Packet A has 200 bytes of payload, packet B 140, packet C 100, packet D 340, and packet E 160. Packets A and C have their marker bit set.

The FEC scheme we use will be with one level as illustrated by Figure 6 in Section 10.1. The protection length L0 = 340 octets.

A redundant coding packetization is used with payload type 100. The payload type of the FEC is assumed to be 127. The first four RED packets, RED #1 through RED #4, each contains an individual media packet, A, B, C, or D, respectively. The FEC data protecting the media data in the first four media packets is generated. The fifth packet, RED #5, contains this FEC data as redundant coding along with media packet E.

RED Packet #1: Media Packet A
RED Packet #2: Media Packet B
RED Packet #3: Media Packet C
RED Packet #4: Media Packet D
RED Packet #5: FEC Packet, Media Packet E

RED packets #1 through #4 will have the structure as shown in Figure 18. The RTP header of the RED packet #1 is as shown in Figure 19, with all the other RED packets in similar format with corresponding sequence numbers and timestamps. The primary encoding block header of the RED packets is as shown in Figure 20.

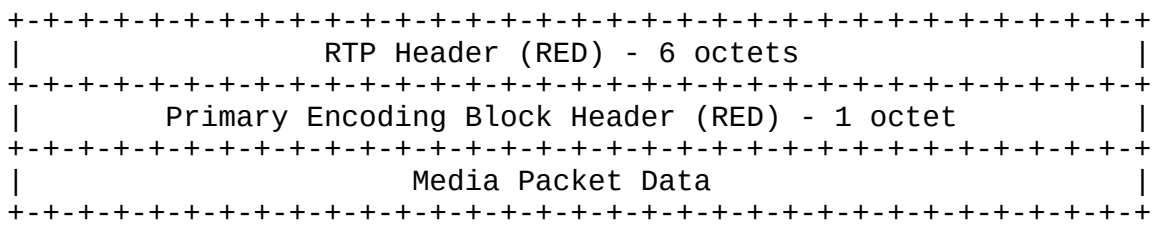
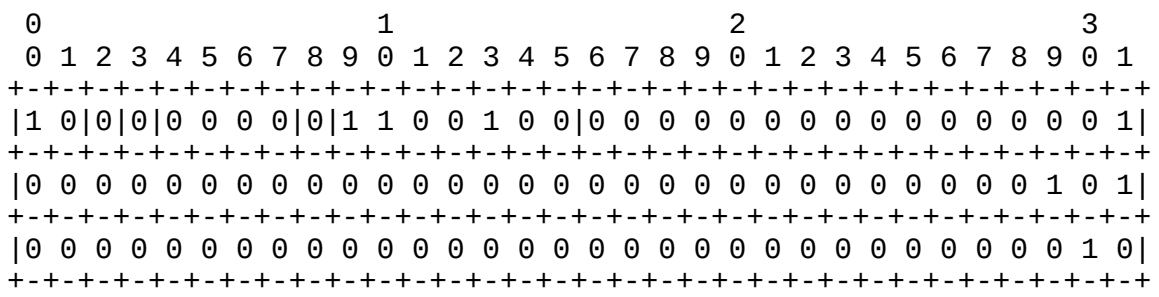
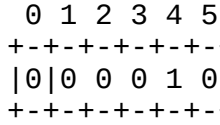


Figure 18: RED Packet Structure - Media Data Only



Version: 2
Padding: 0
Extension: 0
Marker: 0 [Even though media packet A has marker set]
PT: 100 [Payload type for RED]
SN: 1
TS: 5
SSRC: 2

Figure 19: RTP Header of RED Packet #1



F bit: 0 [This is the primary coding data]
Block PT: 11 [The payload type of media]

Figure 20: Primary Encoding Block Header

The FEC data is generated not directly from the RED packets, but from the virtual RTP packets containing the media packet data. Those virtual RTP packets can be very easily generated from the RED packets both with and without redundant coding included. The conversion from RED packets to virtual RTP packets is simply done by (1) removing any RED block headers and redundant coding data, and (2) replacing the PT in the RTP header with the PT of the primary coding.

Note: In the payload format for redundant coding as specified by RFC 2198, the marker bit is lost as soon as the primary coding is carried in the RED packets. So the marker bit cannot be recovered regardless of whether or not the FEC is used.

As mentioned above, RED packet #5 will contain the FEC data (that protects media packets A, B, C, and D) as well as the data of media packet E. The structure of RED packet #5 is as illustrated in Figure 21.

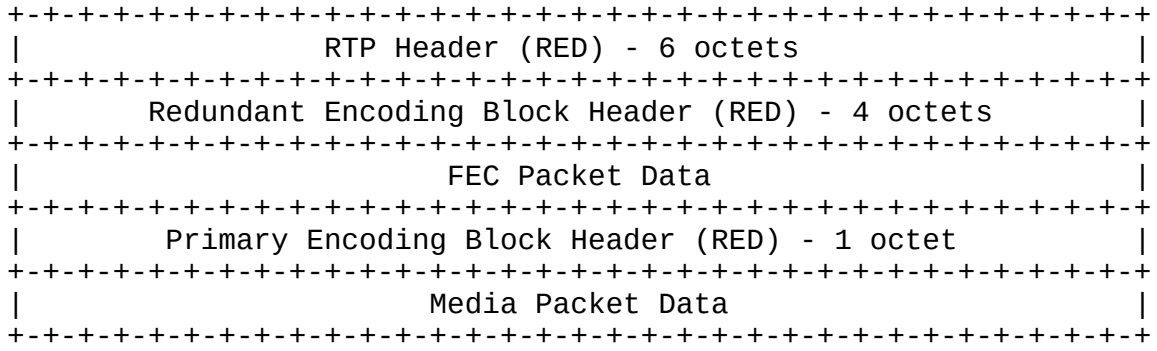
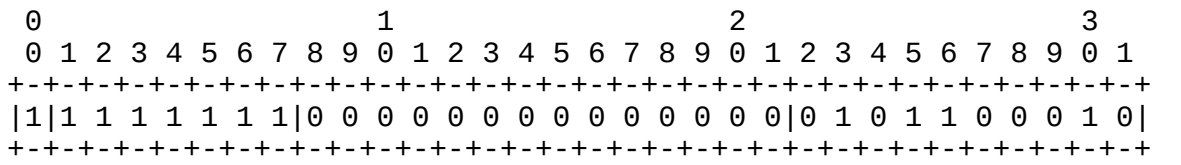


Figure 21: RED Packet Structure - With FEC Data

The RTP header of the RED packets with FEC included is the same as shown in Figure 19, with their corresponding sequence numbers and timestamps.

In RED packet #5, the redundant encoding block header for the FEC packet data block is as shown below in Figure 22. It will be followed by the FEC packet data, which, in this case, includes an FEC header (10 octets as shown in Figure 8), ULP level 0 header (4 octets as shown in Figure 9), and the ULP level 0 data (340 octets as set for level 0). These are followed by the primary encoding block that contains the data of media packet E.




```
   F bit:     1     [This is the redundant coding data]
   Block PT:  127   [The dynamic payload type for FEC]
   TS Offset: 0     [The instance at which the FEC data is
                     transmitted]
   Block Len: 354   [FEC header (10 octets) plus ULP level 0 header
                     (4 octets) and ULP level 0 data (340 octets)]
```

Figure 22: Redundant Encoding Block Header

11. Security Considerations

There are two ways to use FEC with encryption in secure communications: one way is to apply the FEC on already encrypted payloads, and the other way is to apply the FEC before the encryption. The first case is encountered when FEC is needed by a not trusted node during transmission after the media data is encrypted. The second case is encountered when media data is protected by FEC before it is transmitted through a secured transport.

Since the protected payload of this FEC is RTP packets, applying FEC on encrypted payloads is primarily applicable in the case of secure RTP (SRTP) [13]. Because the FEC applies XOR across the payload, the FEC packets should be cryptographically as secure as the original payload. In such cases, additional encryption of the FEC packets is not necessary.

In the following discussion, it is assumed that the FEC is applied to the payload before the encryption. The use of FEC has implications on the usage and changing of keys for encryption. As the FEC packets do consist of a separate stream, there are a number of combinations on the usage of encryption. These include:

- The FEC stream may be encrypted, while the media stream is not.
- The media stream may be encrypted, while the FEC stream is not.
- The media stream and FEC stream are both encrypted, but using the same key.
- The media stream and FEC stream are both encrypted, but using different keys.

The first three of these would require all application-level signaling protocols used to be aware of the usage of FEC, and to thus exchange keys and negotiate encryption usage on the media and FEC streams separately. In the final case, no such additional mechanisms are needed. The first two cases present a layering violation, as ULP FEC packets should be treated no differently than other RTP packets. Encrypting just one stream may also make certain known-plaintext attacks possible. For these reasons, applications utilizing encryption SHOULD encrypt both streams, i.e., the last two options.

Furthermore, because the encryption may potentially be weakened by the known relationship between the media payload and FEC data for certain ciphers, different encryption keys MUST be used for each stream when the media payload and the FEC data are sent in separate streams. Note that when SRTP [13] is used for security of the RTP sessions, different keys for each RTP session are required by the SRTP specification.

The changing of encryption keys is another crucial issue that needs to be addressed. Consider the case where two packets a and b are sent along with the FEC packet that protects them. The keys used to encrypt a and b are different, so which key should be used to decode the FEC packet? In general, old keys need to be cached, so that when the keys change for the media stream, the old key can be used until it is determined that the key has changed for the ULP FEC packets as well. Furthermore, the new key SHOULD be used to encrypt the FEC packets that are generated from a combination of payload packets encrypted by the old and new keys. The sender and the receiver need to define how the encryption is performed and how the keys are used.

Altering the FEC data and packets can have a big impact on the reconstruction operation. An attack by changing some bits in the FEC data can have a significant effect on the calculation and the recovery of the payload packets. For example, changing the length recovery field can result in the recovery of a packet that is too long. Also, the computational complexity of the recovery can easily be affected for up to at least one order of magnitude. Depending on the application scenario, it may be helpful to perform a sanity check on the received payload and FEC data before performing the recovery operation and to determine the validity of the recovered data from the recovery operation before using them.

12. Congestion Considerations

Another issue with the use of FEC is its impact on network congestion. In many situations, the packet loss in the network is induced by congestions. In such scenarios, adding FEC when encountering increasing network losses should be avoided. If it is

used on a widespread basis, this can result in increased congestion and eventual congestion collapse. The applications may include stronger protections while at the same time reduce the bandwidth for the payload packets. In any event, implementations MUST NOT substantially increase the total amount of bandwidth in use (including the payload and the FEC) as network losses increase.

The general congestion control considerations for transporting RTP data apply; see RTP [1] and any applicable RTP profile (e.g., RTP/AVP [14]). An additional requirement if best-effort service is being used is that users of this payload format MUST monitor packet loss to ensure that the packet loss rate is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path, and experiencing the same network conditions, would achieve an average throughput, measured on a reasonable timescale, that is not less than the RTP flow is achieving. This condition can be satisfied by implementing congestion control mechanisms to adapt the transmission rate (or the number of layers subscribed for a layered multicast session), or by arranging for a receiver to leave the session if the loss rate is unacceptably high.

13. IANA Considerations

Four new media subtypes have been registered with IANA, as described in this section. This registration is done using the registration template [3] and following RFC 3555 [4].

13.1. Registration of audio/ulpfec

Type name: audio

Subtype name: ulpfec

Required parameters:

rate: The RTP timestamp rate that is used to mark the time of transmission of the FEC packet in a separate stream. In cases in which it is sent as redundant data to another stream, the rate SHALL be the same as the primary encoding it is used to protect. When used in a separate stream, the rate SHALL be larger than 1000 Hz, to provide sufficient resolution to RTCP operations. The selected rate MAY be any value above 1000 Hz but is RECOMMENDED to match the rate of the media this stream protects.

Optional parameters:

onelevelonly: This specifies whether only one level of FEC protection is used. The permissible values are 0 and 1. If 1 is signaled, only one level of FEC protection SHALL be used in the stream. If 0 is signaled, more than one level of FEC protection MAY be used. If omitted, it has the default value of 0.

Encoding considerations: This format is framed (see Section 4.8 in the template document [3]) and contains binary data.

Security considerations: The same security considerations apply to these media type registrations as to the payloads for them, as detailed in RFC 5109.

Interoperability considerations: none

Published specification: RFC 5109

Applications that use this media type: Multimedia applications that seek to improve resiliency to loss by sending additional data with the media stream.

Additional information: none

Person & email address to contact for further information:
Adam Li adamli@hyervision.com
IETF Audio/Video Transport Working Group

Intended usage: COMMON

Restrictions on usage: This media, type depends on RTP framing, and hence is only defined for transfer via RTP [1]. Transport within other framing protocols SHALL NOT be defined as this is a robustness mechanism for RTP.

Author:
Adam Li adamli@hyervision.com

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

13.2. Registration of video/ulpfec

Type name: video

Subtype name: ulpfec

Required parameters:

rate: The RTP timestamp rate that is used to mark the time of transmission of the FEC packet in a separate stream. In cases in which it is sent as redundant data to another stream, the rate SHALL be the same as the primary encoding it is used to protect. When used in a separate stream, the rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. The selected rate MAY be any value above 1000 Hz, but is RECOMMENDED to match the rate of the media this stream protects.

Optional parameters:

onelevelonly: This specifies whether only one level of FEC protection is used. The permissible values are 0 and 1. If 1 is signaled, only one level of FEC protection SHALL be used in the stream. If 0 is signaled, more than one level of FEC protection MAY be used. If omitted, it has the default value of 0.

Encoding considerations: This format is framed (see Section 4.8 in the template document [3]) and contains binary data.

Security considerations: The same security considerations apply to these media type registrations as to the payloads for them, as detailed in RFC 5109.

Interoperability considerations: none

Published specification: RFC 5109

Applications that use this media type: Multimedia applications that seek to improve resiliency to loss by sending additional data with the media stream.

Additional information: none

Person & email address to contact for further information:
Adam Li adamli@hyervision.com
IETF Audio/Video Transport Working Group

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [1]. Transport within other framing protocols SHALL NOT be defined as this is a robustness mechanism for RTP.

Author:
Adam Li adamli@hyervision.com

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

13.3. Registration of text/ulpfec

Type name: text

Subtype name: ulpfec

Required parameters:

rate: The RTP timestamp rate that is used to mark the time of transmission of the FEC packet in a separate stream. In cases in which it is sent as redundant data to another stream, the rate SHALL be the same as the primary encoding it is used to protect. When used in a separate stream, the rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. The selected rate MAY be any value above 1000 Hz, but is RECOMMENDED to match the rate of the media this stream protects.

Optional parameters:

onelevelonly: This specifies whether only one level of FEC protection is used. The permissible values are 0 and 1. If 1 is signaled, only one level of FEC protection SHALL be used in the stream. If 0 is signaled, more than one level of FEC protection MAY be used. If omitted, it has the default value of 0.

Encoding considerations: This format is framed (see Section 4.8 in the template document [3]) and contains binary data.

Security considerations: The same security considerations apply to these media type registrations as to the payloads for them, as detailed in RFC 5109.

Interoperability considerations: none

Published specification: RFC 5109

Applications that use this media type: Multimedia applications that seek to improve resiliency to loss by sending additional data with the media stream.

Additional information: none

Person & email address to contact for further information:
Adam Li adamli@hyervision.com
IETF Audio/Video Transport Working Group

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [1]. Transport within other framing protocols SHALL NOT be defined as this is a robustness mechanism for RTP.

Author:
Adam Li adamli@hyervision.com

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

13.4. Registration of application/ulpfec

Type name: application

Subtype name: ulpfec

Required parameters:

rate: The RTP timestamp rate that is used to mark the time of transmission of the FEC packet in a separate stream. In cases in which it is sent as redundant data to another stream, the rate SHALL be the same as the primary encoding it is used to protect. When used in a separate stream, the rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. The selected rate MAY be any value above 1000 Hz, but is RECOMMENDED to match the rate of the media this stream protects.

Optional parameters:

onelevelonly: This specifies whether only one level of FEC protection is used. The permissible values are 0 and 1. If 1 is signaled, only one level of FEC protection SHALL be used in the stream. If 0 is signaled, more than one level of FEC protection MAY be used. If omitted, it has the default value of 0.

Encoding considerations: This format is framed (see Section 4.8 in the template document [3]) and contains binary data.

Security considerations: The same security considerations apply to these media type registrations as to the payloads for them, as detailed in RFC 5109.

Interoperability considerations: none

Published specification: RFC 5109

Applications that use this media type: Multimedia applications that seek to improve resiliency to loss by sending additional data with the media stream.

Additional information: none

Person & email address to contact for further information:
Adam Li adamli@hyervision.com
IETF Audio/Video Transport Working Group

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [1]. Transport within other framing protocols SHALL NOT be defined as this is a robustness mechanism for RTP.

Author:
Adam Li adamli@hyervision.com

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

14. Multiplexing of FEC

The FEC packets can be sent to the receiver along with the protected payload primarily in one of two ways: as a separate stream, or in the same stream as redundant encoding. The configuration options MUST be indicated out of band. This section also describes how this can be accomplished using the Session Description Protocol (SDP), specified in RFC 2327 [8].

14.1. FEC as a Separate Stream

When the FEC packets are sent in a separate stream, several pieces of information must be conveyed:

- The address and port to which the FEC is being sent
- The payload type number for the FEC
- Which media stream the FEC is protecting

There is no static payload type assignment for FEC, so dynamic payload type numbers MUST be used. The SSRC of the FEC stream MUST be set to that of the protected payload stream. The association of the FEC stream with its corresponding stream is done by line grouping in SDP [5] with the FEC semantics [6] or other external means.

Following the principles as discussed in Section 5.2 of RFC 3550 [1], multiplexing of the FEC stream and its associated payload stream is usually provided by the destination transport address (network address and port number), which is different for each RTP session. Sending FEC together with the payload in one single RTP session and multiplex only by SSRC or payload type precludes: (1) the use of different network paths or network resource allocations for the payload and the FEC protection data; (2) reception of a subset of the media if desired, particularly for the hosts that do not understand FEC; and (3) receiver implementations that use separate processes for the different media. In addition, multiplexing FEC with payload data streams will affect the timing and sequence number space of the original payload stream, which is usually undesirable. So the FEC stream and the payload stream SHOULD be sent through two separate RTP session, and multiplexing them by payload type into one single RTP session SHOULD be avoided. In addition, the FEC and the payload MUST NOT be multiplexed by SSRC into one single RTP session since they always have the same SSRC.

Just like any media stream, the port number and the payload type number for the FEC stream are conveyed in their m line in the SDP. There is no static payload type assignment for FEC, so dynamic payload type numbers MUST be used. The binding to the number is indicated by an rtpmap attribute. The name used in this binding is "ulpfec". The address that the FEC stream is on is conveyed in its corresponding c line.

The association relationship between the FEC stream and the payload stream it protects is conveyed through media line grouping in SDP (RFC 3388) [5] using FEC semantics (RFC 4756) [6]. The FEC stream and the protected payload stream form an FEC group.

The following is an example SDP for FEC application in a multicast session:

```
v=0
o=adam 289083124 289083124 IN IP4 host.example.com
s=ULP FEC Seminar
t=0 0
c=IN IP4 224.2.17.12/127
a=group:FEC 1 2
a=group:FEC 3 4
m=audio 30000 RTP/AVP 0
a=mid:1
m=application 30002 RTP/AVP 100
a=rtpmap:100 ulpfec/8000
a=mid:2
m=video 30004 RTP/AVP 31
a=mid:3
m=application 30004 RTP/AVP 101
c=IN IP4 224.2.17.13/127
a=rtpmap:101 ulpfec/8000
a=mid:4
```

The presence of two a=group lines in this SDP indicates that there are two FEC groups. The first FEC group, as indicated by the "a=group:FEC 1 2" line, consists of stream 1 (an audio stream using PCM [14]) and stream 2 (the protecting FEC stream). The FEC stream is sent to the same multicast group and has the same Time to Live (TTL) as the audio, but on a port number two higher. The second FEC group, as indicated by the "a=group:FEC 3 4" line, consists of stream 3 (a video stream) and stream 4 (the protecting FEC stream). The FEC stream is sent to a different multicast address, but has the same port number (30004) as the payload video stream.

14.2. FEC as Redundant Encoding

When the FEC stream is being sent as a secondary codec in the redundant encoding format, this must be signaled through SDP. To do this, the procedures defined in RFC 2198 [7] are used to signal the use of redundant encoding. The FEC payload type is indicated in the same fashion as any other secondary codec. The FEC MUST protect only the main codec, with the payload of FEC engine coming from virtual RTP packets created from the main codec data. The virtual RTP packets can be very easily converted from the RFC 2198 packets by simply (1) removing all the additional headers and the redundant coding data, and (2) replacing the payload type in the RTP header with that of the primary codec.

Note: In the payload format for redundant coding as specified by RFC 2198, the marker bit is lost as soon as the primary coding is carried in the RED packets. So the marker bit cannot be recovered regardless of whether or not the FEC is used.

Because the FEC data (including the ULP header) is sent in the same packets as the protected payload, the FEC data is associated with the protected payload by being bundled in the same stream.

When the FEC stream is sent as a secondary codec in the redundant encoding format, this can be signaled through SDP. To do this, the procedures defined in RFC 2198 [7] are used to signal the use of redundant encoding. The FEC payload type is indicated in the same fashion as any other secondary codec. An rtpmap attribute MUST be used to indicate a dynamic payload type number for the FEC packets. The FEC MUST protect only the main codec.

For example:

```
m=audio 12345 RTP/AVP 121 0 5 100
a=rtpmap:121 red/8000/1
a=rtpmap:100 ulpfec/8000
a=fmtp:121 0/5/100
```

This SDP indicates that there is a single audio stream, which can consist of PCM (media format 0), DVI (media format 5), the redundant encodings (indicated by media format 121, which is bound to red through the rtpmap attribute), or FEC (media format 100, which is bound to ulpfec through the rtpmap attribute). Although the FEC format is specified as a possible coding for this stream, the FEC MUST NOT be sent by itself for this stream. Its presence in the m line is required only because non-primary codecs must be listed here according to RFC 2198. The fmtp attribute indicates that the redundant encodings format can be used, with DVI as a secondary coding and FEC as a tertiary encoding.

14.3. Offer / Answer Consideration

Some considerations are needed when SDP is used for offer / answer [15] exchange.

The "onelevelonly" parameter is declarative. For streams declared as sendonly, the value indicates whether only one level of FEC will be sent. For streams declared as recvonly or sendrecv, the value indicates what the receiver accepts to receive.

When the FEC is sent as a separate stream and signaled through media line grouping in SDP (RFC 3388) [5] using FEC semantics (RFC 4756) [6], the offering side MUST implement both RFC 3388 and RFC 4756. The rules for offer / answer in RFC 3388 and RFC 4756 SHALL be followed with the below additional consideration. For all offers with FEC, the answerer MAY refuse the separate FEC session by setting the port to 0, and remove the "a=group" attribute that groups that FEC session with the RTP session being protected. If the answerer accepts the usage of FEC, the answerer simply accepts the FEC RTP session and the grouping in the offer by including the same grouping in the answer. Note that the rejection of the FEC RTP session does not prevent the media sessions from being accepted and used without FEC.

When the FEC stream is sent as a secondary codec in the redundant encoding format (RFC 2198) [7], the offering side can indicate the FEC stream as specified in Section 14.2. The answerer MAY reject the FEC stream by removing the payload type for the FEC stream. To accept the usage of FEC, the answerer must in the answer include the FEC payload type. Note that in cases in which the redundancy payload format [7] is used with FEC as the only secondary codec, when the FEC stream is rejected the redundant encoding payload type SHOULD also be removed.

15. Application Statement

This document describes a generic protocol for Forward Error Correction supporting a wide range of short block parity FEC algorithms, such as simple and interleaved parity codes. The scheme is limited to interleaving parity codes over a distance of 48 packets. This FEC algorithm is fully compatible with hosts that are not FEC-capable. Since the media payload is not altered and the protection is sent as additional information, the receivers that are unaware of the generic FEC as specified in this document can simply ignore the additional FEC information and process the main media payload. This interoperability is particularly important for compatibility with existing hosts, and also in the scenario where many different hosts need to communicate with each other at the same time, such as during multicast.

The generic FEC algorithm specified in this document is also a generic protection algorithm with the following features: (1) it is independent of the nature of the media being protected, whether that media is audio, video, or otherwise; (2) it is flexible enough to support a wide variety of FEC mechanisms and settings; (3) it is

designed for adaptivity, so that the FEC parameters can be modified easily without resorting to out-of-band signaling; and (4) it supports a number of different mechanisms for transporting the FEC packets.

The FEC specified here also provides the user with Unequal Error Protection capabilities. Some other algorithms may also provide the Unequal Error Protection capabilities through other means. For example, an Unequal Erasure Protection (UXP) scheme has been proposed in the AVT Working Group in "An RTP Payload Format for Erasure-Resilient Transmission of Progressive Multimedia Streams". The UXP scheme applies unequal error protection to the media payloads by interleaving the payload stream to be protected with the additional redundancy information obtained using Reed-Solomon operations.

By altering the structure of the protected media payload, the UXP scheme sacrifices the backward compatibility with terminals that do not support UXP. This makes it more difficult to apply UXP when backward compatibility is desired. In the case of ULP, however, the media payload remains unaltered and can always be used by the terminals. The extra protection can simply be ignored if the receiving terminals do not support ULP.

At the same time, also because the structure of the media payload is altered in UXP, UXP offers the unique ability to change packet size independent of the original media payload structure and protection applied, and is only subject to the protocol overhead constraint. This property is useful in scenarios when altering the packet size of the media at transport level is desired.

Because of the interleaving used in UXP, delays will be introduced at both the encoding and decoding sides. For UXP, all data within a transmission block need to arrive before encoding can begin, and a reasonable number of packets must be received before a transmission block can be decoded. The ULP scheme introduces little delay at the encoding side. On the decoding side, correctly received packets can be delivered immediately. Delay is only introduced in ULP when packet losses occur.

Because UXP is an interleaved scheme, the unrecoverable errors occurring in data protected by UXP usually result in a number of corrupted holes in the payload stream. In ULP, on the other hand, the unrecoverable errors due to packet loss in the bitstream usually appear as contiguous missing pieces at the end of the packets. Depending on the encoding of the media payload stream, many applications may find it easier to parse and extract data from a

packet with only a contiguous piece missing at the end than a packet with multiple corrupted holes, especially when the holes are not coincident with the independently decodable fragment boundaries.

The exclusive-or (XOR) parity check operation used by ULP is simpler and faster than the more complex operations required by Reed-Solomon codes. This makes ULP more suitable for applications where computational cost is a constraint.

As discussed above, both the ULP and the UXP schemes apply unequal error protection to the RTP media stream, but each uses a different technique. Both schemes have their own unique characteristics, and each can be applied to scenarios with different requirements.

16. Acknowledgments

The following authors have made significant contributions to this document: Adam H. Li, Fang Liu, John D. Villasenor, Dong-Seek Park, Jeong-Hoon Park, Yung-Lyul Lee, Jonathan D. Rosenberg, and Henning Schulzrinne. The authors would also like to acknowledge the suggestions from many people, particularly Stephen Casner, Jay Fahlen, Cullen Jennings, Colin Perkins, Tao Tian, Matthieu Tisserand, Jeffery Tseng, Mark Watson, Stephen Wenger, and Magnus Westerlund.

17. References

17.1. Normative References

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

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

[3] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[4] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

[5] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, December 2002.

[6] Li, A., "Forward Error Correction Grouping Semantics in Session Description Protocol", RFC 4756, November 2006.

[7] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

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

17.2. Informative References

[9] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.

[10] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", RFC 2354, June 1998.

[11] Rosenberg, J. and H. Schulzrinne, "Registration of parityfec MIME types", RFC 3009, November 2000.

[12] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "Forward Error Correction (FEC) Building Block", RFC 3452, December 2002.

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

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

[15] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

Editor’s Address

Adam H. Li
10194 Wateridge Circle #152
San Diego, CA 92121
USA
Phone: +1 858 622 9038
EMail: adamli@hyervision.com

Full Copyright Statement

Intellectual Property