An RTP Payload Format for Generic Forward Error Correction

STATUS OF THIS MEMO

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited.

1 Abstract

This document specifies a payload format for generic forward error correction of media encapsulated in RTP. It is engineered for FEC algorithms based on the exclusive or (parity) operation, although it can be used with other techniques. The payload format allows end systems to transmit using arbitrary block lengths and parity schemes. It also allows for the recovery of both the payload and critical RTP header fields. It is backwards compatible with non-FEC capable hosts, so that receivers which do not wish to implement FEC can just ignore the extensions.

2 Background

The quality of packet voice on the Internet has been mediocre due, in part, to high packet loss rates. This is especially true on wide-area connections. Unfortunately, the strict delay requirements of real-time multimedia usually eliminate the possibility of retransmissions. It is for this reason that forward error correction (FEC) has been
proposed to compensate for packet loss in the Internet [1] [2]. In particular, the use of traditional error correcting codes, such as parity, Reed-Solomon, and Hamming codes, has attracted attention. To support these mechanisms, protocol support is required.

Budge, McKenzie, Mills, Diss, and Long have proposed a payload format for RTP which allows for the encapsulation of FEC-protected media on top of RTP [3]. We briefly summarize their proposal, and urge the reader to consult their draft for more details. They define a new RTP payload type which identifies the packet contents as FEC-protected media. The RTP payload format in their proposal consists of two elements, the media-correction header and the payload. The media-correction header is 24 bits, and consists of three fields. The first is called the scheme, the second the mode, and the third, the length. The scheme identifies the particular error correction scheme in use. In particular, it defines the set of data packets over which the FEC is applied, and the order in which the packets (data and FEC) are sent. The mode identifies which packet in a group of data and FEC packets (typically called a block) this particular one corresponds to. For packets that contain just data (and not FEC), the length field contains the length of the payload. For packets which contain FEC, the length field contains the xor of the length fields of the packets which are covered by the FEC. Since packets must be padded out with zeroes (to be equal lengths) in order to perform the xor operation, the length field allows recovery of the actual length of the pre-padded packets.

3 Motivation

The payload format proposed in [3] works quite well, but has a number of drawbacks:

- It does not indicate the media type of the actual data being protected. This is because the RTP PT field always indicates that the payload format is "FEC-protected media". Since many applications will need to change media payload types mid-stream (for example, sending DTMF tones in-band), the presence of this field is important.

- The RTP timestamp field and marker bit are not covered by FEC. When a packet is lost and then reconstructed, the timestamp and marker bits are copied from another packet. Correct recovery of these fields is important.

- It defines four very specific schemes (one of which is no error correction), and assigns a value for the scheme field in the header to each. New schemes must be registered with IANA, the details written up, and receivers and senders alike must be
upgraded to recognize and support them. This makes backwards compatibility difficult, requiring capabilities negotiation. It also means that transmitters are restricted to using the schemes defined thus far. The three non-null schemes defined in [3] use heavy forward error correction. These schemes are not appropriate for all loss conditions.

It is our aim to generalize the payload format for forward error protection. To do this, the details of the scheme are transmitted inside the data packets with minimal overhead. This allows sender-based adaptation of the FEC schemes. This adaptation can be static or dynamic, and based on any information available at the sender. Changing schemes mid-stream is then trivially supported, whereas special protocol support is required in [3]. Capability exchanges are avoided, simplifying the protocol and eliminating compatibility problems.

4 Protocol Overview

Before discussing the protocol, we define a few terms for clarity. A media payload is a piece of raw, un-protected user data. A media header is the RTP header for the media payload. The combination of a media payload and media header is called a media packet. The forward error correction algorithms at the transmitter take the media packets as an input. They output both the media packets that they are passed, and new packets called FEC packets. The FEC packet contains an FEC header and FEC payload. Each FEC packet is said to be associated with one or more media packets when those media packets are used to generate the FEC packet (by use of the exclusive or operation, for example).

At the receiver, arriving FEC and media packets are used to generate a stream of media packets for direct use by the application. This results in a clean separation of error protection from the application.

The protocol operates by assuming that the error correction algorithm works by applying some function $f$ to one or more media payloads, which are specified as the arguments to $f$. The result of this function is an FEC payload. When the function is applied to just a single media payload, the result is that media payload ($f(a) = a$). When the function is applied to multiple media payloads, the result is some combination of those payloads (the exclusive or would be defined as: $f(a,b,c) = a \text{xor} b \text{xor} c$). We assume $f$ can combine any number of payloads, each with arbitrary lengths. Recovery is possible if a sufficient number of FEC and media packets are received. Sufficiency depends on the reception of $N$ packets (media or FEC) which contain linearly independent combinations of at most $N$ media packets.
For example, consider the case where f is the exclusive or. Media packets w, x, y, and z, with media payloads a, b, c and d are to be transmitted. Pairs of media payloads will be xor'ed together to generate the FEC payloads. We would denote the resulting network packet stream as:

a, b, f(a,b), c, d, f(c,d)

In this example, the error correction scheme introduces a 50% overhead. But if b is lost, a and f(a,b) can be used to recover b.

The way in which the various schemes differ is in the set of media payloads over which the exclusive-or (or more generally, f(.)) is applied, and the order in which the resulting packets are sent. For example, Budge et. al. describe four schemes, 0, 1, 2, and 3 which take media payloads a, b, c, d, etc., and generate FEC payloads as follows:

Scheme 0
--------

This scheme is null, and has no error correction. The scheme is formally defined as:

a, b, c, d, ... -> a, b, c, d, ....

Scheme 1
--------

This scheme is the similar to the one in the example above. The switching of the positions of f(b) and f(a,b) allow some bursts of two consecutive packet losses to be recovered. It is defined as:

a, b, c, d, e, f -> a, f(a,b), b, f(b,c), c, f(c,d), d, ...

Scheme 2
--------

This scheme allows for recovery of all single packet losses and some consecutive packet losses, but with less overhead than scheme 1:

a, b, c, d, e, f, g -> f(a,b), f(a,c), f(a,b,c), f(c,d), f(c,e), f(c,d,e)...

Scheme 3
This scheme requires 4 packet delays to recover the original media payloads, but it can recover from 1, 2, or 3 consecutive packet losses:

\[ a, b, c, d, e, f \rightarrow f(a), f(b), f(a, b, c), f(c), f(a, c, d), f(a, b, d), f(d), \ldots \]

In order to decode the FEC payloads to media payloads, all that is necessary is for the receiver to know the function being applied, and the set of media payloads in each FEC payload to which it is applied. This is exactly the information provided by the payload format.

To determine the function \( f() \) being used, the payload type for the FEC packets only is set to indicate this function. This payload type may either be static or dynamic, just like any other payload type.

To determine which packets are associated with the FEC packet, a field is present in the FEC header, called the offset mask. Assume this mask is \( M \) bits. If bit \( k \) in the mask is set to 1, then the media packet with sequence number \( N - \text{floor}(M/2) + k \) is associated with this FEC packet, where \( N \) is the sequence number in the FEC packet header. This effectively allows an FEC packet to be associated with any of the \( M/2 \) packets before and after it. For example, if packet 3 is an FEC packet, and it contains the xor of the payload of media packets 1 and 2, and \( M \) is 5, bits 0 and 1 are set to 1, while bits 3, 4, and 5 are set to zero.

The offset mask and payload type are sufficient to signal arbitrary forward error correction schemes with little overhead.

5 Protocol Specifics

The following section fills in the details based on the general discussion above.

5.1 RTP Media Packet Structure

Not all packets transmitted by the source contain FEC. Many contain just regular media information, which would be sent if no error correction were used. The syntax and semantics of the RTP header and payload fields are identical to those defined in RFC 1889 and RFC 1890.

This lends to a very efficient encoding. When little (or no) FEC is used, there are mostly media packets being sent. This means that the overhead (present in FEC packets only) tracks the amount of FEC in
5.2 RTP FEC Packet Structure

When a packet is to be transmitted which contains FEC data (i.e., its payload is derived from one or more media payloads), the RTP header is followed by an FEC header. We first discuss the semantics of the RTP header fields.

The version, padding, extension, CC, SSRC, and CSRC list fields have the normal meaning. Note that in order to be well defined, all packets associated with the FEC packet should have identical values for these fields. The sequence number has the standard definition: it is one higher than the sequence number in the previously transmitted packet. The marker bit is defined as the operator $f()$ applied to the marker bit in the media header of the media packets associated with the FEC packet. In other words, if the FEC packet payload is computed via the exclusive or of the two previous media packet payloads, the marker bit is set to the exclusive or of the marker bit in the two previous media packet headers. This allows the marker bit to be recovered in the same fashion as the payload.

The timestamp field no longer has a well-defined meaning, since the contents of the RTP packets are not generated by a continuous media stream – they now have FEC packets interspersed. One option would be to define this field in the same way as the marker bit is defined. This, however, has two drawbacks:

- It will severely break RTP header compression.
- Since the timestamp field will have an essentially random value, the FEC payload cannot be transmitted using the redundant codec payload format [4].

For these reasons, we instead define the timestamp to be the timestamp of the earliest media packet associated with the FEC packet. This is a somewhat natural definition. More importantly, it fixes the second problem above, although not the first (we note that no value for the TS field will solve this problem if FEC packets are sent separately).

The payload type for the FEC packet is set as described above. End systems which cannot recognize a payload type must discard it anyway. This provides backwards compatibility. The FEC mechanisms can then be used in a multicast group with mixed FEC-capable and FEC-uncapable receivers.

Following the RTP header is the FEC header. This header is nominally 32 bits, and may be optionally extended to 64 bits. The format of the header is as follows:
The length recovery field is used to determine the length of any recovered packets. It is computed by applying \( f \) to the 16 bit natural binary representation of the lengths (in bytes) of the media payloads associated with this FEC packet. This allows for the FEC procedure to be applied even when the lengths of the media packets are not identical. For example, assume an FEC packet is being generated by xor'ing two media packets together. The length of the two media packets are 3 (0b011) and 5 (0b101) bytes, respectively. The length recovery field is then encoded as 0b011 xor 0b101 = 0b110.

The E bit indicates a header extension. When set to 1, it indicates that an additional 32 bits of header follow. The PT recovery field is set to the function \( f() \) applied to the payload types of the media packets associated with the FEC packet. The mask field is either 8 bits or 40 bits, depending on the value of E. If bit \( k \) is set to 1, then the media packet with sequence number \( SN - \text{floor}(M/2) + k \) is associated with this FEC packet, where \( SN \) is the sequence number field in the FEC packet header, and \( M \) is either 8 or 40, depending the value of the E bit.

The payload of the FEC packet is the \( f() \) operator applied to the payloads of the media packets associated with the FEC packet. If the payloads are not of equal length, they must be padded with zeroes to be as long as the longest payload before computing \( f() \).

### 5.3 Recovery Procedures

The FEC packets allow end systems to recover from the loss of media packets. Both the payload type and marker bit of the media packet can be reconstructed. This section describes the procedure for performing this recovery.

Assume a receiver has received several media and FEC packets, but the media packet with sequence number \( xi \) was lost. When an FEC packet arrives, the following steps are taken.

The receiver determines if it has received sufficient packets in order to recover \( xi \). This will be the case if an FEC packet has been received which is associated with \( xi \), and the other packets associated with that FEC packet have either been received or recovered. More complex scenarios are possible as well. For example, if an FEC packet is lost, the receiver may need to rely on other FEC packets to reconstruct the missing media packet.
A xor B xor C, and another containing A xor B have been received, C may be recovered.

Let $T$ be the list of packets (FEC and media) which can be combined to recover $x_i$. The payload of $x_i$ is recovered by applying the inverse of $f$ to the other received payloads in $T$. In the case of xor, this would imply xor’ing the payloads together. This payload may have padding in it. The length of the actual payload is computed via the length recovery field. The $f$ operator is applied to the length recovery fields of the packets in $T$. If one of these packets is not an FEC packet (and thus has no length recovery field), the length recovery field is derived by computing the 16 bit natural binary representation of the payload. The result the application of $f$ is the length of the missing packet. If the payload of $x_i$ is more than this, bits are stripped off from the end so that the length is correct. The marker bit for $x_i$ is computed by applying $f$ to the marker bits of the packets in $T$. The payload type field is computed by applying $f$ to the PT field (for media packets) or the PT recovery field (for FEC packets) of the packets in $T$.

The timestamp for $x_i$ is computed by any reasonable approximation, at the discretion of the implementor. The remaining RTP header fields are copied from any media packet in $T$.

This procedure completely recovers the lost packet, including the payload and RTP header fields.

5.4 Example

Consider 2 media packets to be sent, $x$ and $y$. We wish to protect them by sending one FEC packet which is derived from $x$ and $y$. The $f$ operator is implemented using xor. The three packets are:

**Media Packet x**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>2 (10)</td>
</tr>
<tr>
<td>Padding:</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Extension:</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Marker:</td>
<td>0 (0)</td>
</tr>
<tr>
<td>PTI:</td>
<td>11 (01011)</td>
</tr>
<tr>
<td>SN:</td>
<td>8 (1000)</td>
</tr>
<tr>
<td>TS:</td>
<td>3 (011)</td>
</tr>
<tr>
<td>SSRC:</td>
<td>2 (10)</td>
</tr>
</tbody>
</table>

The payload length is 10 bytes.
Media Packet y
-------------

Version: 2 (10)
Padding: 0 (0)
Extension: 0 (0)
Marker: 1 (1)
PTI: 18 (10010)
SN: 9 (1001)
TS: 5 (101)
SSRC: 2 (10)

The payload length is 11 bytes.

The FEC packet is then:

FEC Packet (contains a xor b)
-----------------------------

Version: 2 (10)
Padding: 0 (0)
Extension: 0 (0)
Marker: 1 (1) (NOTE: 0 xor 1 = 1)
PTI: 191 (NOTE: Assume PTI 191 implies XOR FEC)
SN: 10 (1010)
TS: 3 (11)
SSRC: 2 (10)

len. rec.: 1 (1) (NOTE: 10 xor 11 = 1010 xor 1011 = 0001)
PTI rec.: 24 (11001)
E: 0 (0)
mask: 12 (00001100)

The payload length is 11 bytes.

6 Use with Redundant Encodings

One can consider an FEC packet as a redundant coding of the media. Because of this, the payload format for encoding of redundant audio data [4] can be used to carry the FEC data along with the media. The procedure for this is simple. In some media packet, the payload type
is set to the value for redundant encodings. The secondary coder is then set to be the FEC data. This means that the PTI of the secondary coder is set to the PTI value which indicates FEC. The block length of the secondary coder is set to the length of the FEC header and payload. The timestamp offset is set to the difference between the media timestamp and the timestamp from the FEC packet. The secondary coder payload includes the FEC header and FEC payload.

This procedure only works if an FEC packet is sent after at least one of the media packets it is associated with has been sent. Otherwise, the timestamp offset would be negative, which is not allowed.

Using the redundant encodings payload format also implies that the marker bit cannot be recovered.

An advantage of this approach is a reduction in the overhead for sending FEC packets. It also fixes the RTP compression problem, since the RTP header no longer contains timestamp hiccups from the FEC packet.

7 Open Issues

There are a number of open issues to be resolved. The change in definition of the RTP header fields will affect many of the parameters sent in RTCP packets. For example, jitter computations may have to exclude FEC packets. Octet counts and number of transmitted packets probably should include FEC, however.

8 Conclusion

This draft has presented a new payload format which allows for forward error correction of audio visual media. It is generic, allowing any sender defined error correction schemes to be used which meets the required criteria (any xor based strategy meets the criteria). It is also backwards compatible with existing implementations. Receivers which cannot understand FEC can discard the FEC packets, and still receive the media packets.

9 Security Considerations

There are no security considerations beyond those discussed in [5] and [6].

10 Author’s Addresses

Jonathan Rosenberg
Lucent Technologies, Bell Laboratories
Bibliography


