Internet Engineering Task Force
INTERNET-DRAFT
draft-guillemot-genrtp-01.txt

Audio Visual Transport WG
C.Guillemot, P.Christ, S.Wesner
INRIA / Univ. Stuttgart - RUS
April 19, 1999
Expires: October 18, 1999

RTP Payload for MPEG-4 ES with Scaleable & Flexible Error Resiliency

Status of this Memo

ABSTRACT

This document describes a payload, generic in the sense both of being useable and common for different MPEG-4 elementary streams (ES) types such as audio, video, scene descriptions etc. and providing for the transport of both ES and Sync Layer packet streams.
Relying on fragmentation, grouping and extension data mechanisms which can dynamically adapt to network conditions, the proposed payload allows for protection against loss in a generic way. These mechanisms are supposed to operate both on full and partial Access Units such as PDUs and typed “segments”, the latter units suggesting an extended and possibly normative ESI: These mechanisms both can cover a broad range of protection schemes and additionally do allow to avoid extra connection management complexity – e.g. for separate FEC channels - in high-number-of streams MPEG-4 applications.

1. Introduction

This document is motivated by the large variety of MPEG-4 compressed streams,and of potential error control mechanisms. In addition to a generic unique payload, the motivation here is flexibility in associating error control mechanisms with the compressed media streams, making these error control mechanisms evolutive withoutthe need for redefining the payload format, and adaptable to stream segment types and network characteristics.

Inspired from some genericity concepts from [2-3] on one hand, trying to federate the different error control approaches [1,4, 5, 6] under a unique protocol support mechanism on the other hand, the rationale for this payload proposal consists in:

- genericity with simple, yet sufficient, fragmentation and grouping mechanisms. The same payload could hence be used for MPEG-4 audio, video, and possibly for scene description and object descriptors Elementary Streams (ES), according to their QoS requirements.
- Protection against packet loss, with a generic mechanism, that could, if used, avoid extra connection management complexity possibly brought by separate FEC channels. Indeed, in MPEG-4 applications, the number of streams can potentially be high.
- flexible support of a range of error control mechanisms, from no protection to FEC and redundant data, that could be adapted and applied to typed segments (partial AUs or segments being – in terms of the encoding syntax - syntactical and semantically meaningful parts of an AU – cf. [14], 7.2.3,
  Note: “Such partial AUs may have significance for improved error resilience”.) - and to network characteristics. Redundant data, as in the sense of [1], or of [5-6] (e.g. under the form of repeated picture headers, or of the HEC field of the MPEG-4 video syntax [12]) could then be supported by a unique mechanism.
- Common solution for real-time and non-real-time (streaming from servers) scenarios.
- Unified approach for the transport of both Elementary (ES) and SL-packetized (SPS) Streams.

Some unavoidable 'specificity' is moved from the payload type to an extension field type, that can be signaled also out-of-band a at the beginning of the session, using for example SDP [11].

2. Design Considerations

Figure 2 below shows the adapted model relying on a network adaptation layer that would be both media and network aware. This model allows for a unified solution for real-time and non-real-time (streaming from servers) scenarios.

The compression layer organizes the ESs in Access Units, the smallest entities that can be attributed (explicitly or indirectly) individual timestamps. The compression layer may be network aware (real-time applications) or network unaware (playback from servers).

The compression layer will pass via the ESI to the network adaptation layer full or partial Access Units such as PDUs and typed “segments”, together with indications of AU boundaries, random access points, desired timing information as described by the SLConfigDescriptor. In the given context, the present Version 1 of MPEG-4 enforces normativity through the Sync Layer (SL). An equivalent normative behavior can be assured by the combination of the SLConfigDescriptor (normative anyway) and the ESI. As the present SL is obviously at most appropriate for networks with packet loss (as opposed e.g. to bit errors) as the main quality impairment – it would be much more appropriate to standardize the ESI.

The ESs would be passed to the network adaptation layer together with additionnal parameters as listed in figure 1 below.

```
DTS
CTS
OCR
IdleFlag
      loop( randomAccess Flag
            AUStartFlag
            AUEndFlag
            Esdata
            dataLength
            degradationPriority
            segmentType)
```

Figure 1: An ESI providing typed segments

The media and network aware adaptation layer will support additional fragmentation of AUs if the compression layer is network unaware (playback from servers scenario), will support grouping of partial AUs such as typed segments was well as protection mechanisms for different typed segments which could be adapted to varying network conditions during the session, and possibly to a degradation priority indicated by the compression layer via the ESI.

The protocol support (payload field specification) for fragmentation and grouping is inspired from [2-3] with an attempt for simplification.

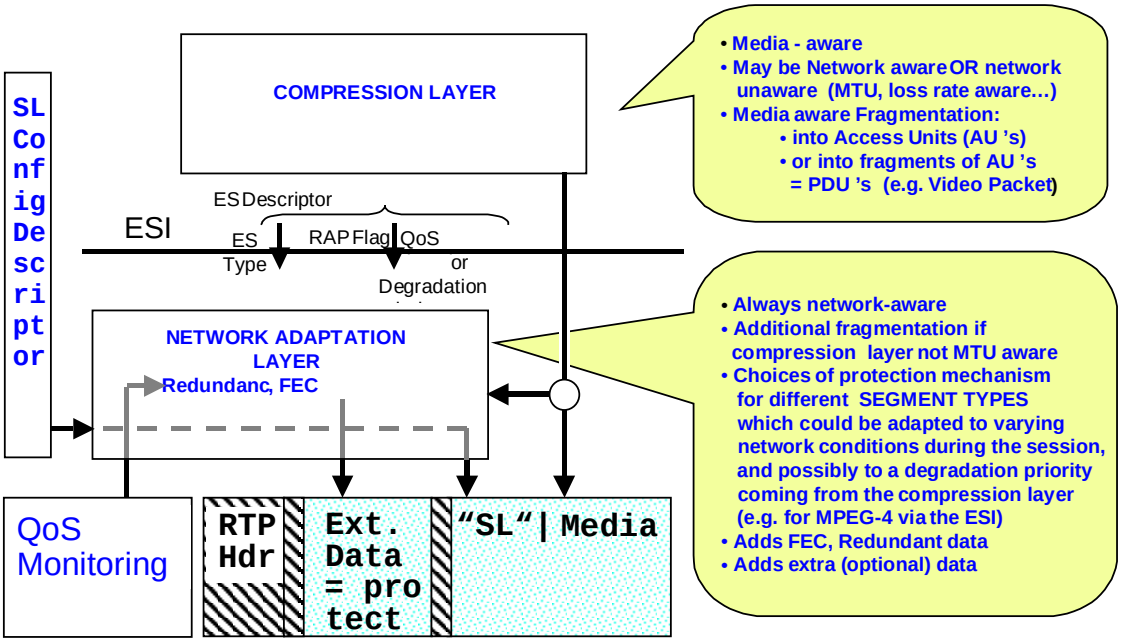


Figure 2: MPEG-4 RTP payload and a network- and media-aware adaptation layer
Legend: "SL" see Appendix A.

Remark: Including error resilience into the compression layer represents a tight coupling between source and channel coding - being optimal only for a specific channel. The model proposed here implements a loosely coupling, the channel coding being dynamically adaptable.

2.1. Fragmentation

The ESs from the compression layer are passed to the network adaptation layer as full AUs or as typed segments or PDUs(partial AUs or segments being – in terms of the
encoding syntax - syntactical and semantically meaningful parts of an AU – cf. [14], 7.2.3; Note: "Such partial AUs may have significance for improved error resilience".)

In the case where the compression layer is not network aware (playback from servers), passed full AUs may have to be fragmented into possibly non independently decodable fragments. This media-unaware fragmentation is however not recommended, passing partial AUs under the form of typed segments is recommended for such applications.

RTP packets transporting fragments or transporting PDUs belonging to the same AU will have their RTP timestamp set at the same value.

2.2. Grouping Mechanisms

The grouping mechanism concerns first the possibility of concatenating several AU's and/or PDUs and/or 'typed segments' in one packet; the grouping mechanism concerns also the possibility of aggregating extension data and PDU in the same packet, as proposed in [2]. Grouping and fragmentation may be combined.

2.3. Unified payload Elementary and SL-packetized Streams

The proposed payload allows the transport of both Elementary (ES) and SL-packetized (SPS) Streams

2.4. Data Characterization

Compressed streams are usually characterized by bit segments containing information with different priority levels, in the sense that the loss of these segments will lead to different impacts, from decoder no-start, to a whole range of quality impairments, including loss of entire frames. Therefore, it seems natural to envisage different levels of protections for stream segments for improved resiliency against packet losses, as proposed in [9] for video, motivating the design of a payload with a flexible support of a range of error control mechanisms that could be adapted to the stream segments types.

The different priority levels considered here are according to three main criteria:

- impact on decoder initialization,
- quality degradation due to packet loss,
- delay requirements.

leading to the consideration, in this document, of the following priority levels or stream segment types:

- HPTD (High Priority with Tolerance to Delay): Vital information, e.g. decoder initialization and configuration that can tolerate increased end-to-end latency (e.g. beginning of the video session, scene description streams such as MPEG-4 BIFS streams, VRML, ...)
- HPND (High Priority with No tolerance to Delay): Vital information (e.g decoder configuration parameters - picture types, quantization values, ...) that cannot tolerate increased delay
- LP (Lower Priority information) (e.g. video frames without configuration parameters,...)

2.4. Hierarchy of Error Control Solutions

Different solutions for increased error resiliency are usually considered, either based on reliable transport protocols for highly sensitive and high priority data [9], either relying on error control mechanisms. Error control mechanisms as ARQ (Automatic Repeat Request) and FEC-based error control mechanisms aim at increasing the stream resiliency to packet loss, but do not avoid packet loss. More precisely, closed loop mechanisms as ARQ techniques consist in re-transmitting the lost data. With open loop mechanisms such as redundant data - which can be repeated data or data encoded with different schemes [1] -, or FEC (Forward Error Correction) - e.g. parity data or data

encoded using block codes -, is transmitted along with the original data, so that some of the lost original data can be recovered from the redundant information.

Reliable transport increases overall latency and delay, which can be incompatible with delay requirements of real-time multimedia, whereas error control mechanisms increase bandwidth as well as overall delay. Also, the potential of the different error control mechanisms, depends on the characteristics of both the packet loss process, and of the compressed media streams.

This motivates the design of a payload that would provide support, with a unique mechanism, for a range of error control solutions, i.e.

- redundant data, with different types (e.g. lower rate secondary data, duplicated 'HPND' information,...)
- FEC, based on parity data or block codes
- no protection

3. Payload Format specification

The packet will consist of an RTP header followed by possibly multiple payloads.

3.1. RTP Header Usage

Each RTP packet starts with a fixed RTP header. The following fields of the fixed RTP header are used:

- Marker bit (M bit): The marker bit of the RTP header is set to 1 when the current packet carries the end of an access unit AU, or the last fragment of an AU.

- Payload Type (PT): The payload type shall be set to value assigned to this format or a payload type in the dynamic range should be chosen.

- Timestamp: The RTP timestamp encodes the presentation time (see appendix A) of the first AU contained in the packet. The RTP timestamp may be the same on successive packets if an AU (e.g. audio or video frame) occupies more than one packet. If the packet contains only 'extension' data objects (see below), then the RTP timestamp is set at the value of the presentation time of the AU to which the first extension data object (e.g. FEC or redundant data) applies.

3.2. Payload Header

The payload header is always present, with a variable length, and is defined as follows:

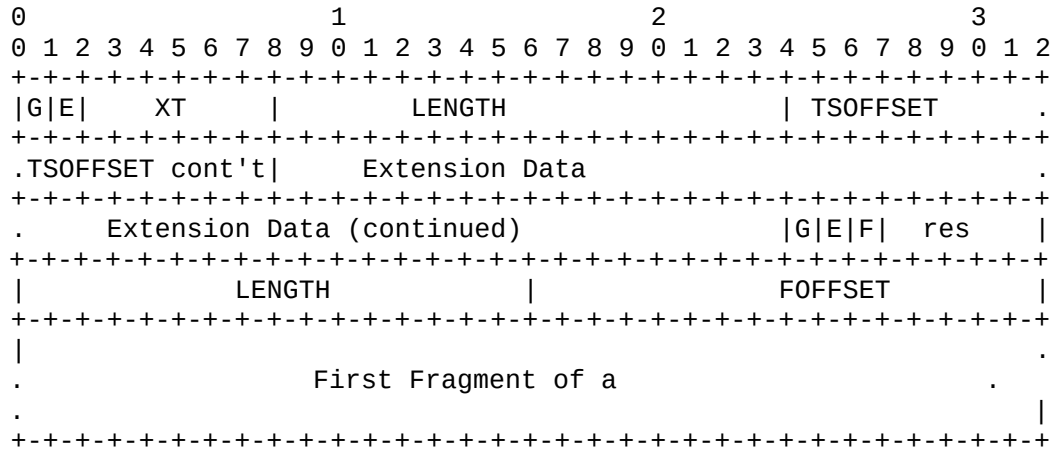


Figure 3. Sample RTP payload, using the payload format.

G (Group) (1 bit): If this field is 1, it indicates that the object associated to the current header is followed by another object.

E (Extension) (1 bit): If its value is 1 then the next object contains Extension data (similarly to [2]). If its value is 0, then the next

F (Fragmentation) (1 bit): This field is only present when the E-field is 0 and the G-field is 0. If its value is 1, then the next 'object field' is a fragment of a PDU. If this field is 0, then the next 'object field' is a complete PDU.

res (Reserved) (5 bits): this field is only present if the E-field is 0, resulting in always 1 byte for {G,E=1,XT} or {G,E=0,F,res}

XT (Extension type) (6 bits): This field is only present if E is set to 1. It then specifies the type of extension data. Examples of types will be FEC data with the specification of the FEC coding scheme (parity codes, block codes such as Reed Solomon codes, ...), redundant data with the specification of the redundant data encoding scheme, duplicated high priority data,...etc.

LENGTH (16 bits): this field specifies the length in bytes of the next 'object field'. If the object contains the last PDU or last PDU fragment of the payload then this field is not present.

FOFFSET (16 bits): This field is present only when the F field is present and F=1. It contains the byte offset of the first byte of the fragment from the beginning of the PDU.

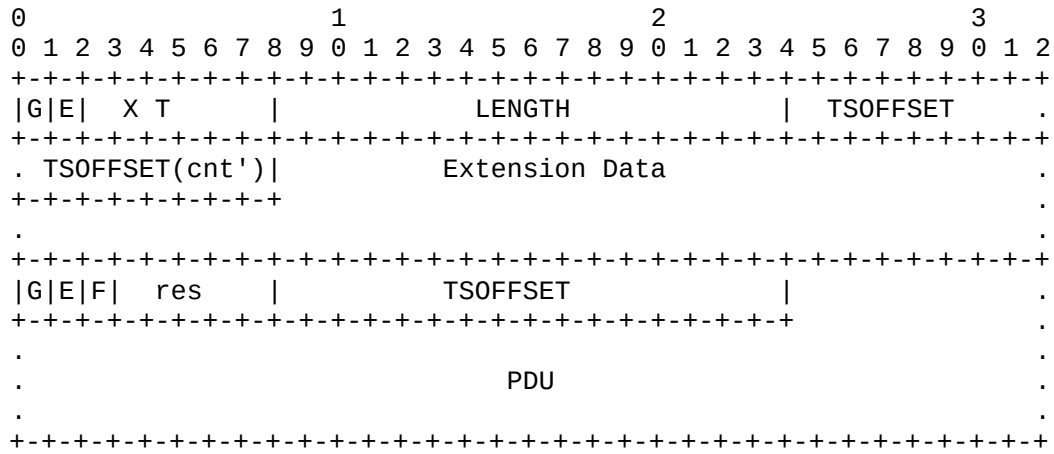
TSOFFSET (Time Stamp OFFSET) (16 bits): The value of the field is an unsigned 16 bit integer. The default value is 0. If the E field is '1', then the next 'object' carries extension data, and the TSOFFSET added to the value of the RTP timestamp yields the presentation time of the PDU to which the extension data apply. The TSOFFSET is, in this case set to the difference between the media TS and the TS of the media to which the extension data

apply. If the E field is '0', then the next 'object field' is a PDU. If this PDU is not the first PDU in the payload (i.e. the previous object is also a PDU), then the TSOFFSET added to the value of the RTP timestamp yields the presentation time of the following PDU. If this PDU or fragment of a PDU is the first in the payload (even if it has been preceded by extension data) then thisfield is not present.

4. Examples of payload headers

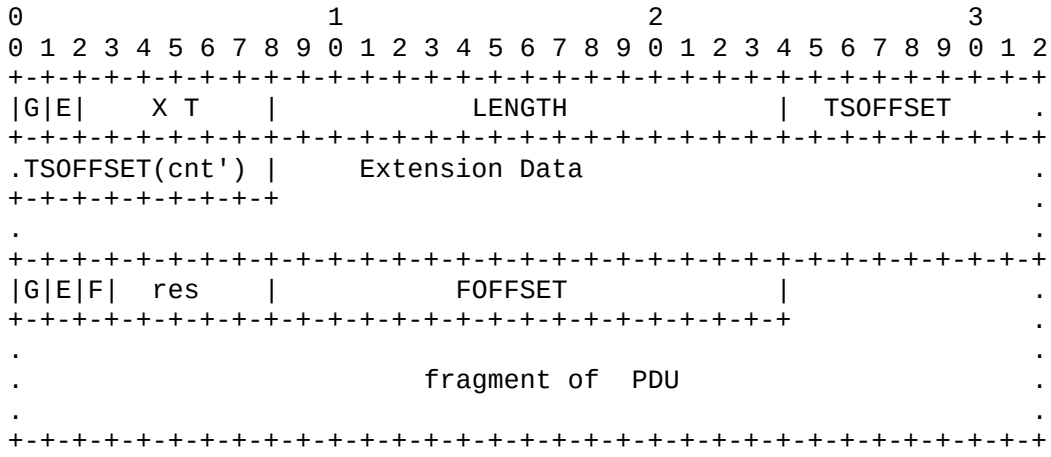
4.1. The payload contains Extension data followed by a PDU

First payload header: G=1, E=1, so F not present, FOFFSET not present;
Second payload header: G=0, E=0, F=0, XT not present, res present, FOFFSET not present (F=0).
last PDU (G=0) in the payload, so the length field is not present.



4.2. The payload contains Extension data followed by a fragment

First payload header: G=1, E=1, so F not present
Second payload header: G=0, E=0, F=1, XT not present, res present,
last fragment (G=0) in the payload, so LENGTH not present.
first fragment in the payload, so TSOFFSET is not present.

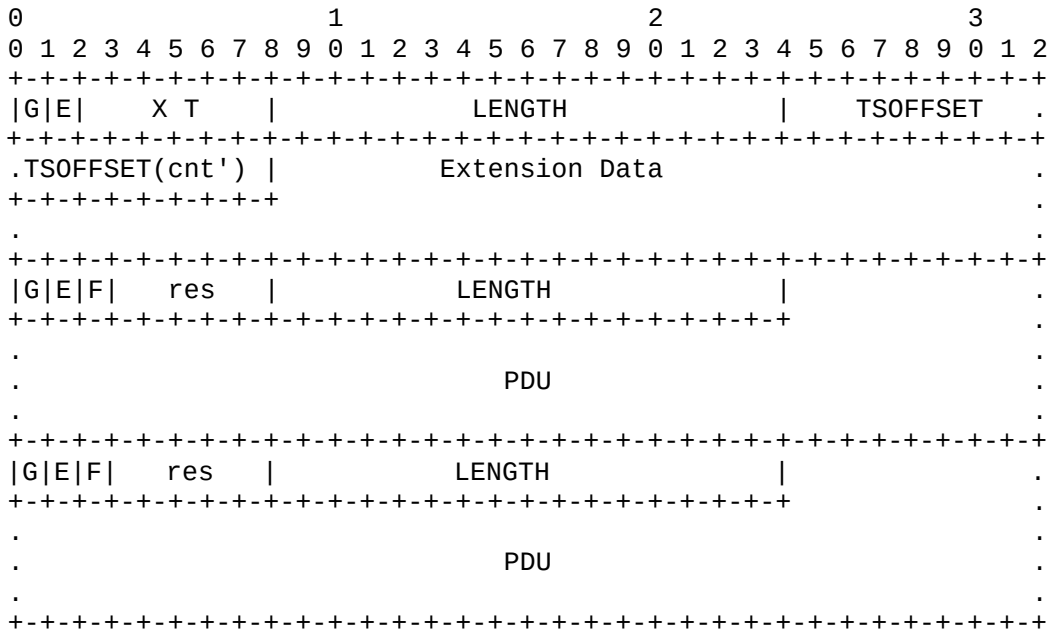


4.3. The payload contains Extension data followed by 2 PDU's

First payload header: G=1, E=1, so F field not present

Second payload header: G=1, E=0, F=0, XT not present, res present, first PDU in the payload, so TSOFFSET is not present.

Third payload header: G=0, E=0, F=0, XT field not present, last PDU in the payload, so LENGTH field not present



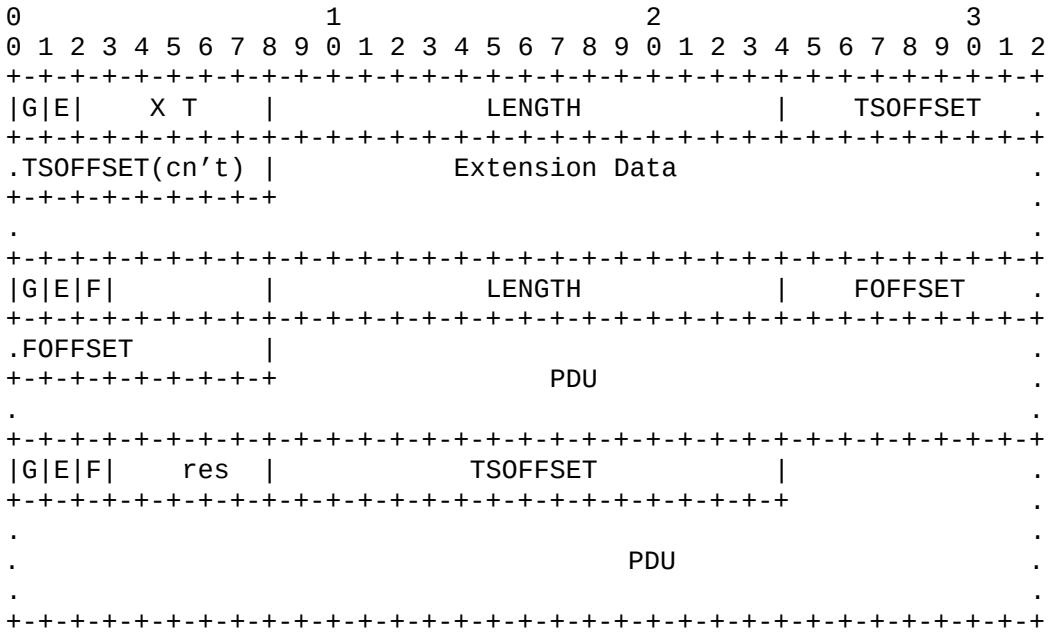
4.4. The payload contains Extension data followed by one fragment followed by one PDU

First payload header: G=1, E=1, so F field not present

Second payload header: G=1, E=0, F=1, XT not present,

first PDU fragment in the payload, so TSOFFSET is not present.

Third payload header: G=0, E=0, F=0, XT field not present, res present, last PDU in the payload, so LENGTH field not present

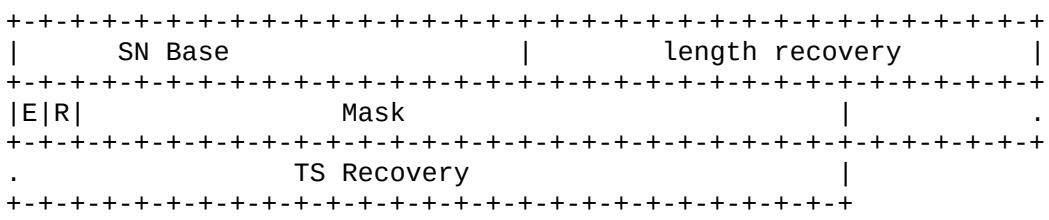


5. Extension data field for parity and Reed Solomon codes

The Extension data field can be used for transporting FEC data in the spirit of [4], but in the same channel as the media data. Similarly to [4], it can support a variety of FEC mechanisms (parity codes, block codes such as Reed Solomon codes). In the approach proposed here, provided that the XT field semantic is announced via a non-RTP and out of band signaling, such as SDP [9], with appropriate extensions, then the FEC mechanisms can, during the session, and depending on the segment type, and on the network characteristics, be adapted without further out of band signaling.

5.1. Parity codes

Inspired from [4], in the case of parity codes, the extension data field can include the following header:



The fields SN base, E, Mask, TS recovery are defined as in [4].

R: The bit R is the Marker recovery bit. The marker bit is computed from the RTP media packets marker bits M, to which is applied the protection operation.

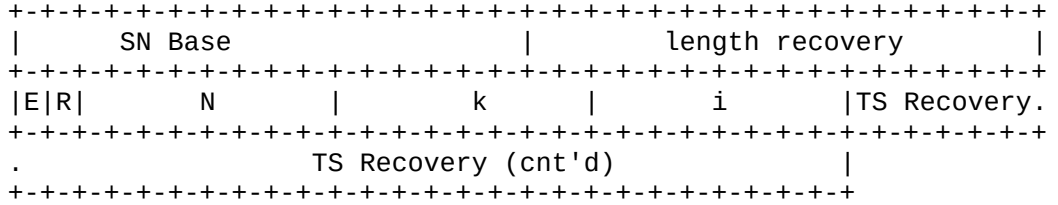
Length recovery:
determines the length of the recovered packets and is here computed via the protection operation applied to the 16 bit

natural binary representation of the lengths (in bytes) of the media payload, CSRC list, extension and padding of media packets associated with this FEC data, PLUS THE MARKER BIT.

The length recovery field allows to apply the procedure to packets which are not of the length, including here to some objects of the given packets.

5.2. Reed Solomon Codes

Similarly, in the case of Reed Solomon codes, the extension data field can include the following header:



Note that, unlike [4], the PT recovery field is not used, since the payload type of the packets transported in a given channel is supposed to be known, namely to be of the type corresponding to this proposed payload.

6. Usage of Extension data field for redundant data

6.1. Usage for redundant 'HPND' data

All AU-level or frame-level decoder configuration information can be considered as of HPND type. This information is of high priority, since, if lost, the whole frame is lost and does not tolerate increased latency.

The extension data field may hence hold duplicated HPND data in 'n' consecutive packets. The parameter 'n' may be chosen so that the probability that 'n' consecutive packets are lost is below a given threshold. But these decision mechanisms are outside the scope of this document.

Note that this duplication of frame-level information is envisaged in payload formats defined for compressed video streams [5-7], in the MPEG-4 video syntax itself, by using a 'Header extension' (HEC) – see annex A -,... with conceptually closed, yet with

6.2. Usage for redundant 'LP' data

As a special type of FEC, it has been proposed in [1] to use, lower rate, secondary encoding of the media data to be protected. The mechanism described above is directly useable for the transport of secondary compressed streams along with primary compressed data. Note that the secondary compressed stream can also be the a lower layer (with a lower rate) of a scaleable compression scheme, such as specified in [12] and [13] for respectively video and audio.

7. Conclusion

This document describes a solution for a 'generic' payload, i.e. unique for a large variety of compressed streams - audio, video, with different compression schemes, or scene description. It allows to have a generic support for flexible error protection which can be in addition adaptable to stream segment characteristics, as well as to network characteristics. Unavoidable 'specificity' is moved from the payload type to the extension data type. The extension data field can be used for supporting mechanisms for improved packet loss resiliency. This concept, with respect to payload headers which are fixed and dedicated to given compression schemes, brings additional flexibility in error resiliency, from no protection (the extension data field will not be used for FEC, redundant data or duplicated headers) to various degrees of protection depending on the types of the following segments of data. An out-of-band mechanism, such as SDP, could be used for announcing at the beginning of the session the semantic of extension data and/or list of extension data types supported. During the session, different extension data types (e.g. for supporting different error protection mechanisms) can then selected and announced without out-of-band mechanisms.

8. Authors Addresses

Christine Guillemot
INRIA
Campus Universitaire de Beaulieu
35042 RENNES Cedex, FRANCE
email: Christine.Guillemot@irisa.fr

Paul Christ
Computer Center - RUS University of Stuttgart
Allmandring 30
D70550 Stuttgart, Germany.
email: Paul.Christ@rus.uni-stuttgart.de

Stefan Wesner
Computer Center - RUS University of Stuttgart
Allmandring 30
D70550 Stuttgart, Germany.
email: wesner@rus.uni-stuttgart.de

9. Bibliography

[1] : C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J. Bolot, A. Vega-Garcia, S. Fosse-Parisis, 'RTP Payload for Redundant Audio Data', draft-ietf-avt-redundancy-revised-00.txt, 10-Aug-98

[2] : A. Klemets, 'Common Generic RTP Payload Format', draft-klemets-generic-rtp-00, March 13, 1998.

[3] : A. Periyannan, D. Singer, M. Speer,'Delivering Media Generically over RTP', draft-periyannan-generic-rtp-00, March 13, 1998

[4] : J. Rosenberg, H. Schulzrinne, 'An RTP Payload Format for Generic Forward Error Correction', draft-ietf-avt-fec-03.txt, 10-Aug-98.

[5] : T. Turletti, C. Huitema, 'RTP payload for H.261 video', RFC 2032.

[6] : C. Zhu, 'RTP payload format for H.263 Video Streams', RFC 2190.

[7] : C. Borman, L. Cline, G. Deisher, T. Gardos, C. Maciocco, D. Newell, J. Ott, S. Wenger, C. Zhu, 'RTP payload format for the 1998 version of ITU-T Rec. H.263 video (H.263+)', draft-ietf-avt-rtp-h263-video-02.txt, 7-May-98.

[8] : D. Hoffman, G. Fernando, V. Goyal, M. Civanlar,'RTP Payload format for MPEG1/MPEG2 video', RFC 2250, January 1998.

[9] : M. Reha Civanlar, G.L. Cash, B.G. Haskell, 'AT&T Error Resilient Video Transmission Technique', draft-civanlar-hplp-00.txt, July 1998.

[10] : H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, 'RTP: A Transport Protocol for Real-Time Applications', draft-ietf-avt-rtp-new-00.ps, December 1997.

[11] : Mark Handley, Van Jacobson, 'SDP:Session Description Protocol', draft-ietf-mmusic-sdp-07.txt, 2nd Apr 1998.

[12] : Information Technology - Coding of Audiovisual Objects, Visual, ISO/IEC 14496-2, May 15, 1998.

[13] : Information Technology - Coding of Audiovisual Objects, Audio - CELP, ISO/IEC 14496-3, subpart 3, May 15, 1998.

[14] : Information Technology - Coding of Audiovisual Objects, ISO/IEC 14496-1: 1999(E), Part 1: Systems, Date: 1998-03-08.

[15] : M. Reha Civanlar et. al., ' RTP Payload Format for MPEG-4 Streams', work in progress, draft-ietf-avt-rtp-mpeg4-01.txt

Appendix A: Sync Layer

In the payload packetization and depacketization process the SLConfigDescriptor is assumed to be accessible.

A.1 Composition Time (CTS)

The presentation time mentioned above is either the Composition Time (CTS) or if its absence is indicated by the SLConfigDescriptor, the RTP packetizer should insert the sampling instant of the first AU in the packet.

A.2. Decoding time (DTS)

If its presence is indicated via the SLConfigDescriptor it should be put in front of the corresponding media payload; corresponding length values are provided by the SLConfigDescriptor.

A.3. Object Clock Reference (OCR)

If its presence is indicated via the SLConfigDescriptor it should be put in front of the corresponding media payload; corresponding length values are provided by the SLConfigDescriptor.

A.4. Sequence numbers

If this payload format is used to accommodate SL-packet streams, the AU-sequence number if present can be placed in front of the media payload; corresponding length values are provided by the SLConfigDescriptor.

A.5. Extension Data do also apply to elements in front of media payload.

Appendix B: Restrictions

B.1. Timestanmp length: If compositionTimeStamp has more than 32 bits length, this payloadformat cannot be used.

Acknowledgements

We want to thank especially Anders Klemets for his initial proposal and helpful comments.