Internet Engineering Task Force
Internet Draft

Avaro-France Telecom
Basso-AT&T
Casner-Packet Design
Civanlar-AT&T
Gentric-Philips
Herpel-Thomson
Lifshitz-Optibase
Lim-mp4cast
Perkins-ISI
van der Meer-Philips

Document: draft-gentric-avt-mpeg4-multisl-00.txt
December 2000
Expires June 2000

RTP Payload Format for MPEG-4 Streams

Status of this Memo

Abstract

This document describes a payload format for transporting MPEG-4 encoded data using RTP. MPEG-4 is a recent standard from ISO/IEC for the coding of natural and synthetic audio-visual data. Several services provided by RTP are beneficial for MPEG-4 encoded data transport over the Internet. Additionally, the use of RTP makes it possible to synchronize MPEG-4 data with other real-time data types.

This specification is a product of the Audio/Video Transport working group within the Internet Engineering Task Force and ISO/IEC MPEG-4

ad hoc group on MPEG-4 over Internet. Comments are solicited and

should be addressed to the working group's mailing list at rem-conf@es.net and/or the authors.

1. Introduction

MPEG-4 is a recent standard from ISO/IEC for the coding of natural and synthetic audio-visual data in the form of audiovisual objects that are arranged into an audiovisual scene by means of a scene description [1][2][3][4]. This draft specifies an RTP [5] payload format for transporting MPEG-4 encoded data streams.

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 [6].

The benefits of using RTP for MPEG-4 data stream transport include:

i. Ability to synchronize MPEG-4 streams with other RTP payloads

ii. Monitoring MPEG-4 delivery performance through RTCP

iii. Combining MPEG-4 and other real-time data streams received from multiple end-systems into a set of consolidated streams through RTP mixers

iv. Converting data types, etc. through the use of RTP translators.

1.1 Overview of MPEG-4 End-System Architecture

Fig. 1 below shows the general layered architecture of MPEG-4 terminals. The Compression Layer processes individual audio-visual media streams. The MPEG-4 compression schemes are defined in the ISO/IEC specifications 14496-2 [2] and 14496-3 [3]. The compression schemes in MPEG-4 achieve efficient encoding over a bandwidth

ranging from several Kbps to many Mbps. The audio-visual content compressed by this layer is organized into Elementary Streams (ESs). The MPEG-4 standard specifies MPEG-4 compliant streams. Within the constraint of this compliance the compression layer is unaware of a specific delivery technology, but it can be made to react to the characteristics of a particular delivery layer such as the path-MTU or loss characteristics. Also, some compressors can be designed to be delivery specific for implementation efficiency. In such cases the compressor may work in a non-optimal fashion with delivery technologies that are different than the one it is specifically designed to operate with.

The hierarchical relations, location and properties of ESs in a presentation are described by a dynamic set of Object Descriptors (ODs). Each OD groups one or more ES Descriptors referring to a

single content item (audio-visual object). Hence, multiple alternative or hierarchical representations of each content item are possible.

ODs are themselves conveyed through one or more ESs. A complete set of ODs can be seen as an MPEG-4 resource or session description at a stream level. The resource description may itself be hierarchical, i.e. an ES conveying an OD may describe other ESs conveying other ODs.

The session description is accompanied by a dynamic scene description, Binary Format for Scene (BIFS), again conveyed through one or more ESs. At this level, content is identified in terms of audio-visual objects. The spatio-temporal location of each object is defined by BIFS. The audio-visual content of those objects that are synthetic and static are described by BIFS also. Natural and animated synthetic objects may refer to an OD that points to one or more ESs that carry the coded representation of the object or its animation data.

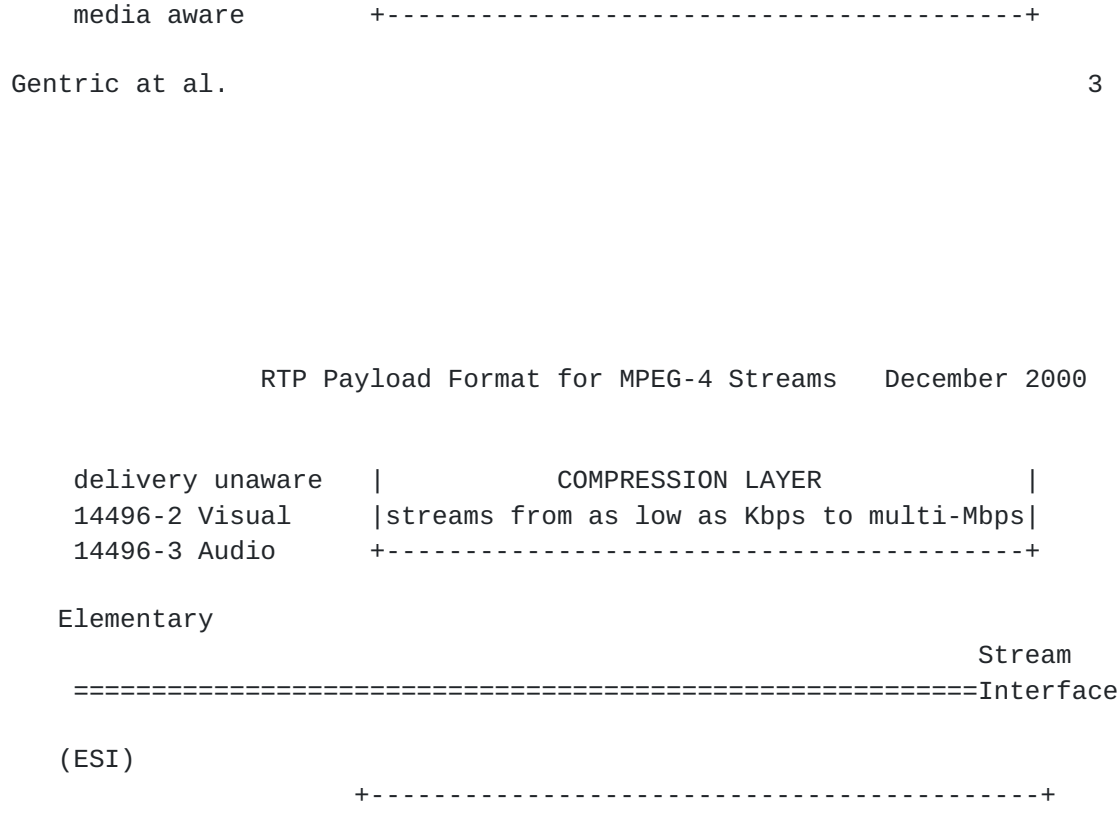
By conveying the session (or resource) description as well as the scene (or content composition) description through their own ESs, it is made possible to change portions of the content composition and the number and properties of media streams that carry the audio-

visual content separately and dynamically at well known instants in time.

One or more initial Scene Description streams and the corresponding OD stream has to be pointed to by an initial object descriptor (IOD). The IOD needs to be made available to the receivers through some out-of-band means that are not defined in this document.

A homogeneous encapsulation of ESs carrying media or control (ODs, BIFS) data is defined by the Sync Layer (SL) that primarily provides the synchronization between streams. The Compression Layer organizes the ESs in Access Units (AU), the smallest elements that can be attributed individual timestamps. Integer or fractional AUs are then encapsulated in SL packets. All consecutive data from one stream is called an SL-packetized stream at this layer. The interface between the compression layer and the SL is called the Elementary Stream Interface (ESI). The ESI is informative.

The Delivery Layer in MPEG-4 consists of the Delivery Multimedia Integration Framework defined in ISO/IEC 14496-6 [4]. This layer is media unaware but delivery technology aware. It provides transparent access to and delivery of content irrespective of the technologies used. The interface between the SL and DMIF is called the DMIF Application Interface (DAI). It offers content location independent procedures for establishing MPEG-4 sessions and access to transport channels. The specification of this payload format is considered as a part of the MPEG-4 Delivery Layer.



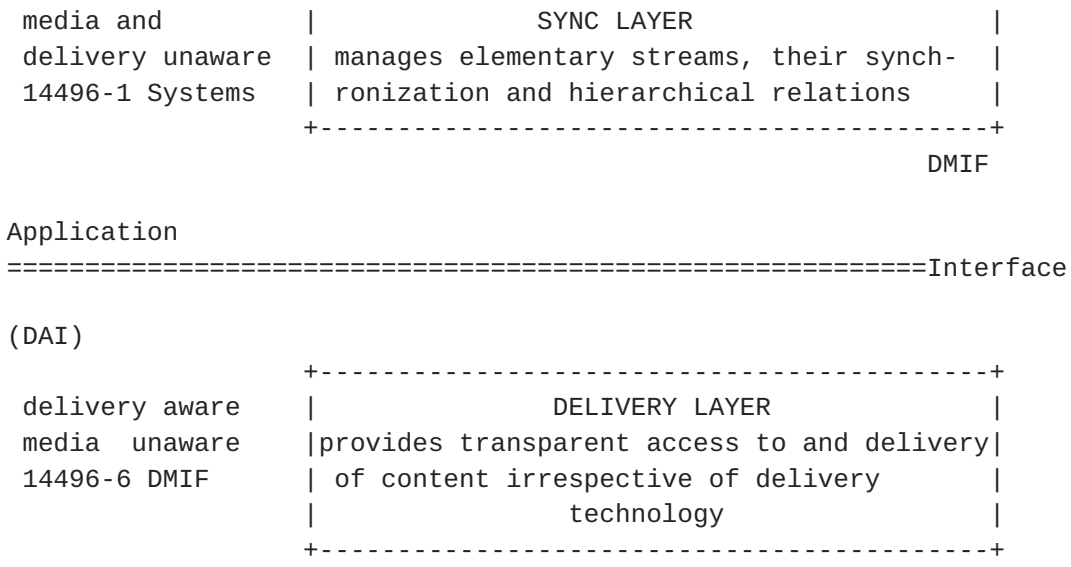


Figure 1: General MPEG-4 terminal architecture

1.2 MPEG-4 Elementary Stream Data Packetization

The ESs from the encoders are fed into the SL with indications of AU boundaries, random access points, desired composition time and the current time.

The Sync Layer fragments the ESs into SL packets, each containing a header that encodes information conveyed through the ESI. If the AU is larger than a SL packet, subsequent packets containing remaining parts of the AU are generated with subset headers until the complete AU is packetized.

The syntax of the Sync Layer is not fixed and can be adapted to the needs of the stream to be transported. This includes the possibility to select the presence or absence of individual syntax elements as well as configuration of their length in bits. The configuration for each individual stream is conveyed in a SLConfigDescriptor, which is an integral part of the ES Descriptor for this stream.

2. Analysis of the alternatives for carrying MPEG-4 over IP

2.1 MPEG-4 over UDP

Considering that the MPEG-4 SL defines several transport related functions such as timing, sequence numbering, etc., this seems to be the most straightforward alternative for carrying MPEG-4 data over IP. One group of problems with this approach, however, stems from the monolithic architecture of MPEG-4. No other multimedia data stream (including those carried with RTP) can be synchronized with MPEG-4 data carried directly over UDP. Furthermore, the dynamic scene and session control concepts can't be extended to non-MPEG-4 data.

Even if the coordination with non-MPEG-4 data is overlooked, carrying MPEG-4 data over UDP has the following additional shortcomings:

i. Mechanisms need to be defined to protect sensitive parts of MPEG-4 data. Some of these (like FEC) are already defined for RTP.

ii. There is no defined technique for synchronizing MPEG-4 streams from different servers in the variable delay environment of the Internet.

iii. MPEG-4 streams originating from two servers may collide (their sources may become unresolvable at the destination) in a multicast session.

iv. An MPEG-4 back channel needs to be defined for quality feedback similar to that provided by RTCP.

v. RTP mixers and translators can't be used.

The back-channel problem may be alleviated by developing a reception reporting protocol like RTCP. Such an effort may benefit from RTCP design knowledge, but needs extensions.

2.2 RTP header followed by full MPEG-4 headers

This alternative may be implemented by using the send time or the composition time coming from the reference clock as the RTP timestamp.
This way no new feedback protocol needs to be defined for MPEG-4's back channel, but RTCP may not be sufficient for MPEG-4's feedback requirements that are still in the definition stage. Additionally, due to the duplication of header information, such as the sequence numbers and time stamps, this alternative causes unnecessary increases in the overhead. Scene description or dynamic session control can't be extended to non-MPEG-4 streams also.

2.3 MPEG-4 ESs over RTP with individual payload types

This is the most suitable alternative for coordination with the existing Internet multimedia transport techniques and does not use MPEG-4 systems at all. Complete implementation of it requires definition of potentially many payload types, as already proposed

for audio and video payloads [7], and might lead to constructing new session and scene description mechanisms. Considering the size of the work involved which essentially reconstructs MPEG-4 systems, this may only be a long term alternative if no other solution can be found.

2.4 RTP header followed by a reduced SL header

The inefficiency of the approach described in 2.2 can be fixed by using a reduced SL header that does not carry duplicate information following the RTP header.

2.5 Recommendation

Based on the above analysis, the best compromise is to map the MPEG-4 SL packets onto RTP packets, such that the common pieces of the headers reside in the RTP header that is followed by an optional reduced SL
header providing the MPEG-4 specific information. The details of this
payload format are described in the next section.

3. Payload Format

The RTP Payload corresponds to an integer number of SL packets. The SLPacket headers are transformed into reduced SL packet headers, with some fields replaced by those in the RTP header and others transported in reduced form. The payload is unchanged.

When generating SL packetized stream specifically for this format all other fields in the SL packet headers that the RTP header does not duplicate (including the decodingTimeStamp) is OPTIONAL.

The packet structure consists in a concatenated header section where

all reduced SL packet headers are bit-wise concatenated. If the resulting concatenated header section consumes a non-integer number of bits zero padding bits MUST be inserted at the end in order to achieve byte-alignment.

After the concatenated header section is the concatenated payload section where all SLPacket payloads are concatenated. SLPacket payloads are byte aligned.

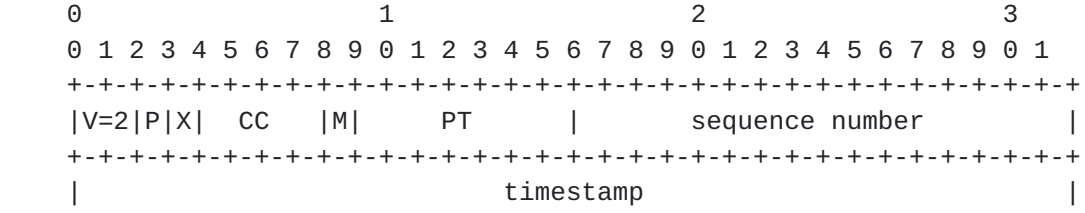
RTP Packets SHOULD be sent in the decoding (MPEG-4 decodingTimeStamp) order.

SL packets inside RTP packets MUST be in the decoding (MPEG-4 decodingTimeStamp) order.

The size of the SL packet(s) SHOULD be adjusted such that the resulting RTP packet is not larger than the path-MTU. To handle larger packets, this payload format relies on lower layers for fragmentation, which may not be desirable.

It is assumed that the MPEG-4 SLConfigDescriptor is transported "out of
band". This is typically done via an ObjectDescriptorStream using the
MPEG-4 Object Description framework.

However since some knowledge of the SLConfigDescriptor is required by
an RTP receiver in order to parse MPEG-4 System specific elements in the RTP payload defined in this document, the SLConfigDescriptor MAY be transported in the SDP associated with such a stream using the a=fmtp syntax (see below).



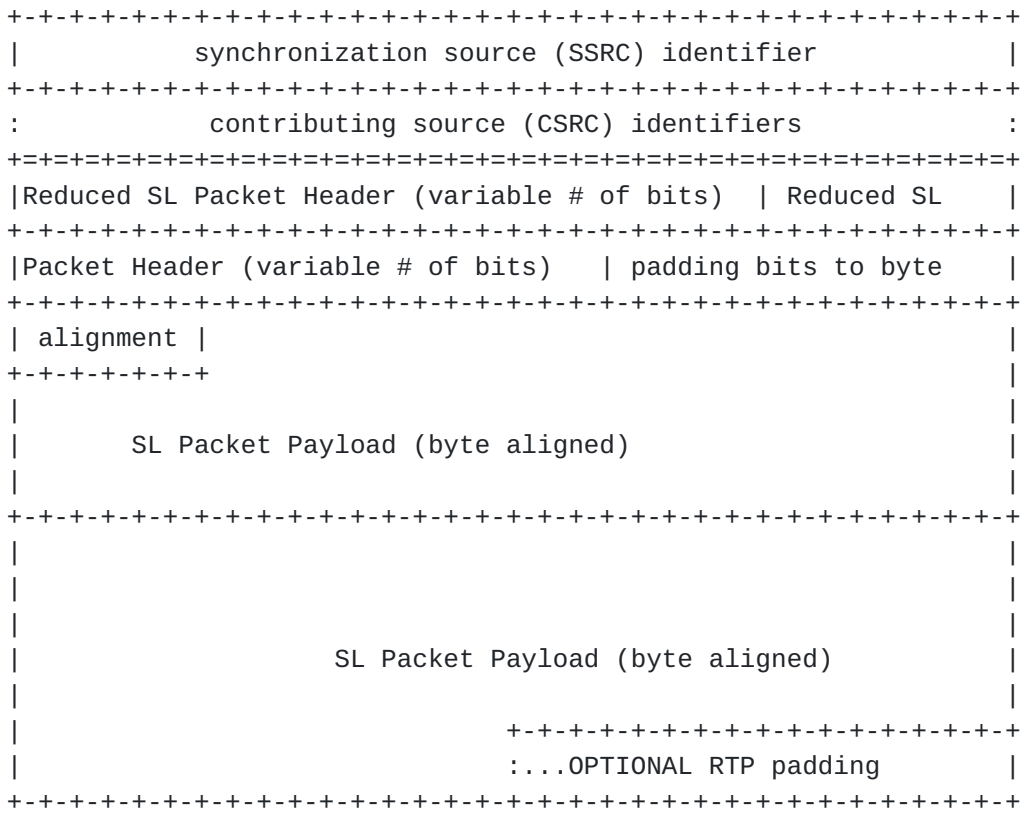


Figure 2: An RTP packet for MPEG-4

3.1 RTP Header Fields Usage

Payload Type (PT): The assignment of an RTP payload type for this new packet format is outside the scope of this document, and will not be specified here. It is expected that the RTP profile for a particular class of applications will assign a payload type for this

encoding, or if that is not done then a payload type in the dynamic range shall be chosen.

Marker (M) bit: Set to one to mark the last fragment (or only fragment) of an AU. Also set to one for RTP packets that

Extension (X) bit: Defined by the RTP profile used.

Sequence Number: The RTP sequence number should be generated by the sender with a constant random offset and does not have to be correlated to any (optional) MPEG-4 SL sequence numbers.

Timestamp: Set to the value in the compositionTimeStamp field of the first SL packet, if present. If compositionTimeStamp has less than 32 bits length, the MSBs of timestamp MUST be set to zero.

Although it is available from the SL configuration data, the resolution of the timestamp may need to be conveyed explicitly through some out-of-band means to be used by network elements which are not MPEG-4 aware.

If compositionTimeStamp has more than 32 bits length, this payload format cannot be used.

In all cases, the sender SHALL always make sure that RTP time stamps are identical only for RTP packets transporting fragments of the same Access Unit.

In case compositionTimeStamp is not present in the current SL packet, but has been present in a previous SL packet the reason is that this is the same Access Unit that has been fragmented therefore the same timestamp value MUST be taken as RTP timestamp.

According to RFC1889 [5, Section 5.1] timestamps are recommended to start at a random value for security reasons. However then, a receiver is not in the general case able to reconstruct the original MPEG-4 Time Stamps (CTS, DTS, OCR) which can be of use for applications where streams from multiple sources are to be synchronized. Therefore the usage of such a random offset SHOULD be avoided.

SSRC: set as described in RFC1889 [5]. A mapping between the ES identifiers (ESIDs) and SSRCs should be provided through out-of-band means.

CC and CSRC fields are used as described in RFC 1889 [5].

RTCP SHOULD be used as defined in RFC 1889 [5].

Reduced SL Header Packet: Defined in section 3.2 and 3.3. If the Reduced SL Packet Header contains a non-integer number of bytes, trailing padding bits, each coded as zero, MUST be inserted to byte align the start of the SL Packet Payload.

SL Packet Payload: The payload of an SL Packet. The payload MUST be byte aligned, if needed, by using trailing padding bits, each coded as zero.

RTP timestamps in RTCP SR packets: according to the RTP timing model, the RTP timestamp that is carried into an RTCP SR packet is the same as the compositionTimeStamp that would be applied to an RTP packet for data that was sampled at the instant the SR packet is being generated and sent. The RTP timestamp value is calculated from the NTP timestamp for the current time, which also goes in the RTCP SR packet. To perform that calculation, an implementation needs to periodically establish a correspondence between the CTS value of a data packet and the NTP time at which that data was sampled.

3.2 Reduced SL Packet header construction

The following modifications of the SL packet header MUST be applied to
a SL packetized stream before encapsulation in this RTP payload format.
The other fields of the SL packet header MUST remain unchanged (but are bit-shifted to fill in the gaps left by the changes specified below).

3.2.1 Time Stamps transformation

The first reduced SL packet includes a header without compositionTimeStamp field since the RTP time stamp transports it. After placing its value in the RTP time stamp, the sender MUST remove the compositionTimeStamp, if any, from the first SL packet header. All other MPEG-4 Time Stamps are encoded as offsets.

If compositionTimeStamp is never present in SL packets for this stream, the RTP packetizer SHOULD convey a reading of a local clock at the time the RTP packet is created.

All decodingTimeStamps, if present, MUST be replaced by the difference between their value and the value of the compositionTimeStamp. If an OCR (Object Clock Reference) is present it MUST also be changed to encode a difference from the compositionTimeStamp in the same fashion. With this payload format OCRs MUST have the same clock resolution as Time Stamps. If compositionTimeStamp is not present for a SL packet that has OCR

then the OCR SHALL be encoded as a difference to the RTP time stamp.

Since this subtraction may lead to negative values, the offset MUST be encoded as a two's complement signed integer in network byte order.

Because these offsets (delta) typically require fewer bits to be encoded, the sender MAY use a different length than the one indicated by the original SLConfigDescriptor timeStampLength field.

The length MUST then be signaled to the receiver by using an SDP a=fmtp field (see section 3.3 and section 8).

3.2.2 Indication of size

For efficiency SL packets do not carry their own size. This is not an issue for RTP packets that contain a single SL Packet.

However when multiple SL packets are carried in a RTP packet the size of each SL packet payload MUST be available to the receiver.

If the SL packet payload size is constant for a stream, the size information SHOULD NOT be transported in the RTP packet. However in that case it MUST be signaled in SDP using a (a=fmtp:<format> SLPacketPayloadSize=<value>) syntax (see section 8).

If the SL packet payload size is variable then the size of each SL packet payload MUST be indicated in the corresponding Reduced SL packet header. In order to do so the reduced SL packet header MUST contain a SLPacketPayloadSize field. Since this field serves the same purpose as the accessUnitLength field, it replaces it, if present, i.e. senders MUST remove accessUnitLength from the original SL packet headers. The number of bits on which this size is described MUST be indicated in the corresponding SDP using a (a=fmtp:<format> SLPacketPayloadSizeLength=<value>) syntax (see section 8).

The absence of either SLPacketPayloadSize or SLPacketPayloadSizeLength in SDP indicates that a single SL packet is transported in each RTP packet for that stream.

3.2.3 Interleaving

SL packets MAY be interleaved.

When interleaving of SL packets is used it SHALL be implemented using PacketSequenceNumber.

Note that AUSequenceNumber in the SL header is not available for interleaving since it may collide with BIFS Carousel usage.

The conjunction of RTP sequence number and packetSequenceNumber can produce a quasi-unique identifier for a SL packet so that a receiver can unambiguously reconstruct the original order even in case of out-of-order packets, packet loss or duplication.

If packetSequenceNumber is used it SHALL be unchanged for the first SL packet but MAY be encoded as a difference for the other SL packets in the same RTP packet. In that case the length in bits on which these packetSequenceNumber differences (delta) are encoded MUST be signaled in SDP using a (a=fmtp:<format> packetSequenceNumberDeltaLength=<value>) syntax (see section 8).

3.2.4 Constraints for use of fields in the remainingSLPacketHeader

3.2.4.1 Random Access Points

Access Units that have the Random Access Point set to true (1) are referred to as RAP.

In case multiple Access Units are transported in a RTP packet, this packet may contain either no RAP or one or more RAPs.

In case one or more RAPs are present the first SL packet MUST be a RAP; the reason being that a receiver after a packet loss may have to skip packets until a RAP and this is facilitated when only the first Reduced SL packet header has to be scanned.

3.2.4.2 Degradation priority

For streams that use the optional degradation priority field in the SL packet headers, only SL packets with the same degradation priority SHALL be transported by one RTP packet so that components may dispatch the RTP packets according to appropriate QOS or protection scheme. Furthermore only the first reduced SL packet header SHALL carry the degradationPriority field since it would be otherwise redundant.

3.2.4.3 AUSequenceNumber

If AUSequenceNumber is used it SHALL be unchanged for the first SL packet but MAY be encoded as a difference for the others SL packets in the same RTP packet. In that case the length in bits on which these AUSequenceNumber differences (delta) are encoded MUST be signaled in SDP using a (a=fmtp:<format> AUSequenceNumberDeltaLength=<value>) syntax (see section 8).

3.3 Reduced SL Packet Headers

The reduced SL Packet Header is configurable and depends on SDP parameters.

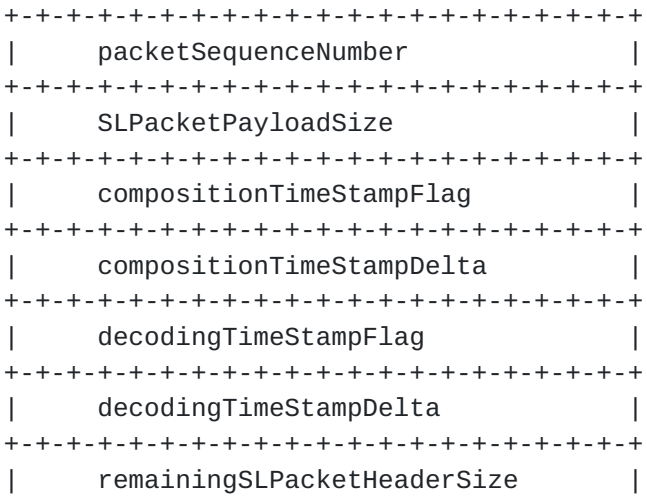


Figure 3: Reduced SL Packet Header

3.3.1 Usage of fields

packetSequenceNumber : Indicates the serial number of the SLPacketPayload. The length of the packetSequenceNumber field is defined by SDP parameters as follows. For the first reduced SL Packet Header in an RTP packet, the length is defined by the packetSequenceNumberLength, and for any subsequent reduced SL Packet Header by the packetSequenceNumberDeltaLength.

SLPacketPayloadSize : Indicates the size in bytes of the associated SL Packet Payload.

compositionTimeStampFlag : Indicates whether the compositionTimeStampDelta field is present. A value of 1 indicates that the field is present, a value of 0 that it is not present.

CompositionTimeStampDelta : Specifies the value of the CTS as a 2-complement offset from the timestamp in the RTP header of this packet.

decodingTimeStampFlag : Indicates whether the decodingTimeStampDelta field is present. A value of 1 indicates that the field is present, a value of 0 that it is not present. If the decodingTimeStampFlag is true, the sender MUST remove the decodingTimeStamp from the original SL packet headers.

DecodingTimeStampDelta : Specifies the value of the DTS as a 2-complement offset from the timestamp in the RTP header of this packet.

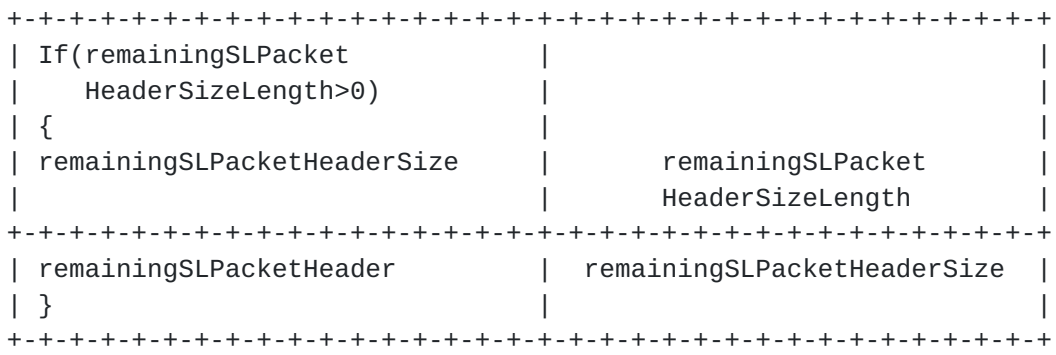
remainingSLPacketHeaderSize : Specifies the length in bits of the immediately following remainingSLPacketHeader.

remainingSLPacketHeader : The remainder of an SL header after removal of the CTS and DTS, field, if any, and modification of the associated flags. The semantics of the original SL Packet Header is defined by a SLConfigDescriptor conveyed in SDP or by other means. If the remaining SL Packet header contains an OCR, then this field is not coded as defined in such descriptor, but instead as described in 3.1.1 with a length indicated by the OCRDeltaLength parameter at SDP. Similarly, if the remaining SL Packet header of a subsequent (non-first) reduced SL Header in an RTP packet contains the AU_sequenceNumber field, then this field may not be coded as defined in such descriptor but instead as described in 3.1.7 with a length indicated by the AUSequenceNumberDeltaLength parameter at SDP.

3.3.2 Relationship between reduced SL Packet header and SDP parameters

The relationship between a reduced SL Packet Header and the SDP parameters is as follows:

Fields of Reduced SL Packet Header	Number of bits
If(packetSequenceNumberLength>0) {	
packetSequenceNumber	packetSequenceNumberLength
}	
If(SLPacketPayloadSizeLength>0) {	
SLPacketPayloadSize	SlpacketPayloadSizeLength
}	
If(decodingTime StampDeltaLength>0) {	
decodingTimeStampFlag	1
If(decodingTimeStampFlag==1) {	
decodingTimeStampDelta	decodingTimeStampDeltaLength
}	
}	
If(compositionTime StampDeltaLength>0) {	
compositionTimeStampFlag	1
If(compositionTimeStampFlag==1) {	
CompositionTimeStampDelta	compositionTimeStampDeltaLength
}	
}	



4. SL packetized stream reconstruction

The MPEG-4 over IP framework [9] requires that the way a receiver can reconstruct a valid SL packetized stream shall be documented, this is the purpose of this section.

Since this format directly transports SL packets this reconstruction is trivial with the following rules:
- The SL packet header SHALL remain exactly the same as received with the following exceptions:
- All time stamps (CTS, DTS, OCR), if present, are restored from the delta values.
- All sequence numbers (packetSequenceNumber, AUSequenceNumber), if present are restored from the delta values relative to the first SL packet in the RTP packet.
- AccessUnitLength fields, if present (i.e. if SL.AU_Length is non zero), are restored from SLPacketPayloadSize.
- The other SL packet header fields SHALL remain exactly the same as in the remainingSLPacketHeader.

5. Multiplexing

Since a typical MPEG-4 session may involve a large number of objects, that may be as many as a few hundred, transporting each ES as an individual RTP session may not always be practical. Allocating and controlling hundreds of destination addresses for each MPEG-4 session may pose insurmountable session administration problems. The input/output processing overhead at the end-points will be

extremely high also. Additionally, low delay transmission of low bitrate data streams, e.g. facial animation parameters, results in extremely high header overheads.

To solve these problems, MPEG-4 data transport requires a multiplexing scheme that allows selective bundling of several ESs. This is beyond the scope of the payload format defined here. MPEG-4's Flexmux multiplexing scheme may be used for this purpose by defining an additional RTP payload format for "multiplexed MPEG-4 streams." Another approach may be to develop a generic RTP multiplexing scheme usable for MPEG-4 data. The multiplexing scheme reported in [8] may be a candidate for this approach.

For MPEG-4 applications, the multiplexing technique needs to address the following requirements:

i. The ESs multiplexed in one stream can change frequently during a session. Consequently, the coding type, individual packet size and temporal relationships between the multiplexed data units must be handled dynamically.

ii. The multiplexing scheme should have a mechanism to determine the ES identifier (ES_ID) for each of the multiplexed packets. ES_ID is not a part of the SL header.

iii. In general, an SL packet does not contain information about its size. The multiplexing scheme should be able to delineate the multiplexed packets whose lengths may vary from a few bytes to close to the path-MTU.

6. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [5]. This implies that confidentiality of the media streams is achieved by encryption. Because the data compression used with this payload format is applied end-to-end, encryption may be performed on the compressed data so there is no conflict between the

two operations. The packet processing complexity of this payload type does not exhibit any significant non-uniformity in the receiver side to cause a denial-of-service threat.

However, it is possible to inject non-compliant MPEG streams (Audio, Video, and Systems) to overload the receiver/decoder's buffers which might compromise the functionality of the receiver or even crash it. This is especially true for end-to-end systems like MPEG where the buffer models are precisely defined.

MPEG-4 Systems supports stream types including commands that are executed on the terminal like OD commands, BIFS commands, etc. and programmatic content like MPEG-J (Java(TM) Byte Code) and ECMASCRIPT. It is possible to use one or more of the above in a manner non-compliant to MPEG to crash or temporarily make the receiver unavailable.

Authentication mechanisms can be used to validate of the sender and the data to prevent security problems due to non-compliant malignant MPEG-4 streams.

A security model is defined in MPEG-4 Systems streams carrying MPEG-J access units which comprises Java(TM) classes and objects. MPEG-J defines a set of Java APIs and a secure execution model. MPEG-J content can call this set of APIs and Java(TM) methods from a set of Java packages supported in the receiver within the defined security model. According to this security model, downloaded byte code is forbidden to load libraries, define native methods, start programs, read or write files, or read system properties.

Receivers can implement intelligent filters to validate the buffer requirements or parametric (OD, BIFS, etc.) or programmatic (MPEG-J, ECMAScript) commands in the streams. However, this can increase the complexity significantly.

7. Types and names

The encoding name associated to this RTP payload format is "mpeg4-sl".

The media type may be any of:
- "video"
- "audio"
- "application"

"video" SHOULD be used for MPEG-4 Video streams (ISO/IEC 14496-2) or MPEG-4 Systems streams that convey information needed for an audio/visual presentation.

"audio" SHOULD be used for MPEG-4 Audio streams (ISO/IEC 14496-3) or MPEG-4 Systems streams that convey information needed for an audio-only presentation.

"application" SHOULD be used for MPEG-4 Systems streams (ISO/IEC 14496-1) that serve other purposes than audio/visual presentation, e.g. in some cases when MPEG-J streams are transmitted.

8. Additional SDP syntax

8.1 Mapping information

This format may require additional information about the mapping to be made available to the receiver.

For example as mentioned above some fields of the SL packet header MAY be reconfigured for optimal efficiency. When such a change is performed however it MUST be signaled to the receiver using a SDP (a=fmtp) parameter as in RFC 2327 [10, section 6].

The absence of any of these fields is similar to a field set to the default value (zero).

8.1.1 Indication of decodingTimeStamp delta bit length

The following syntax should be used:

a=fmtp:<format> decodingTimeStampDeltaLength=<value>

<value> being the number of bits on which the decoding time stamp deltas are encoded in the reduced SL packet headers. The default value is zero. A value larger than zero indicates that the decodingTimeStampFlag is contained in each Reduced SL Packet Header. A value of zero indicates that the decodingTimeStampFlag is not present; in that case, the sender MUST remove any decodingTimeStampFlag from the original SL packet headers.

8.1.2 Indication of compositionTimeStamp delta bit length

The following syntax should be used:

a=fmtp:<format> compositionTimeStampDeltaLength=<value>

<value> being the number of bits on which the composition time stamp deltas are encoded in the (non-first) reduced SL packet headers

8.1.3 Indication of OCR delta bit length

The following syntax should be used:

a=fmtp:<format> OCRDeltaLength=<value>

<value> being the number of bits on which the Object Clock Reference deltas are encoded in the remaindingSLPacketHeader.

8.1.4 Indication of SLPayloadSize description length

The following syntax should be used:

a=fmtp:<format> SLPacketPayloadSizeLength=<value>

<value> being the number of bits on which the SLPacketPayloadSize are encoded in the reduced SL packet headers.

Simultaneous presence in SDP of this parameter and SLPacketPayloadSize is illegal.

8.1.5 Indication of packetSequenceNumber length

The following syntax should be used:

a=fmtp:<format> packetSequenceNumberLength=<value>

<value> being the number of bits on which the packetSequenceNumber is encoded in the first reduced SL packet headers. The default value is zero and indicates the absence of packetSequenceNumber and packetSequenceNumberDelta for all reduced SL headers.

8.1.6 Indication of packetSequenceNumber delta length

The following syntax should be used:

a=fmtp:<format> packetSequenceNumberDeltaLength=<value>

<value> being the number of bits on which the packetSequenceNumberDelta are encoded in any reduced SL packet header subsequent to the first one. If this parameter is not present and the packetSequenceNumberLength parameter is present, then the packetSequenceNumber in any reduced SL header is encoded with the number of bits defined by the value of packetSequenceNumberLength.

8.1.7 Indication of the length in bits of the remainingSLPacketHeaderSize field

The following syntax should be used:

a=fmtp:<format> remainingSLPacketHeaderSizeLength=<value>

<value> being the number of bits that is used to encode the subsequent remaindingSLPacketHeaderSize field. The default value is zero and indicates the absence of the remaindingSLPacketHeaderSize and the remaindingSLPacketHeader fields.

8.1.8 Indication of AUSequenceNumber delta length

The following syntax should be used:

a=fmtp:<format> AUSequenceNumberDeltaLength=<value>

<value> being the number of bits on which the AUSequenceNumberDelta are encoded in the remainderSLPacketHeader. The default value is zero and indicates that AUSequenceNumber, if present, is unchanged in the remaining SL packet header.

8.1.9 Indication of constant SL packet size

The following syntax should be used:

a=fmtp:<format> SLPacketPayloadSize=<value>

<value> being the constant size in bytes of each SL packet payload.

Simultaneous presence in SDP of this parameter and SLPacketPayloadSizeLength is illegal.

8.2 Optional configuration information

In the MPEG-4 framework the following information is carried using the Object Descriptor. For compatibility with receivers that do not implement the full MPEG-4 system specification this information MAY also be indicated in SDP.

For transport of MPEG-4 audio and video without the use of MPEG-4 systems, as well as to support non-MPEG-4 system receivers, it is possible to transport information on the profile and level of the stream and on the decoder configuration.

8.2.1 Indication of SLConfigDescriptor

Senders MAY transmit the SLConfigDescriptor in SDP.

The following syntax should be used:

a=fmtp:<format> SLConfigDescriptor=<value>

<value> being a base-64 encoding of the SLConfigDescriptor. This SHALL be the original SLConfigDescriptor and it SHALL be the same as the one transported by the OD framework.

8.2.2 Indications for MPEG-4 audio streams

8.2.2.1 Indication of profile level

Senders MAY transmit the profile and level indication in SDP.

The following syntax should be used:

a=fmtp:<format> profile-level-id=<value>

<value> being a decimal representation of the MPEG-4 Audio Profile

Level indication value defined in ISO/IEC 14496-1. This parameter indicates which MPEG-4 Audio tool subsets are applied to encode the audio stream.

8.2.2.2 Indication of audio object type

Senders MAY transmit the audio object type indication in SDP.

The following syntax should be used:

a=fmtp:<format> object-type=<value>

<value> being a decimal representation of the MPEG-4 Audio Object Type value defined in ISO/IEC 14496-3. This parameter specifies the tool used by the encoder. It CAN be used to limit the capability within the specified "profile-level-id".

8.2.2.3 Indication of audio bitrate

Senders MAY transmit the audio bitrate in SDP.

The following syntax should be used:

a=fmtp:<format> bitrate=<value>

<value> being a decimal representation of the audio bitrate in bits per second for the audio bit stream.

8.2.2.4 Indication of audio decoder configuration

Senders MAY transmit the audio decoder configuration in SDP.

The following syntax should be used:

a=fmtp:<format> config=<value>

<value> being a hexadecimal representation of an octet string that expresses the audio payload configuration data "StreamMuxConfig", as defined in ISO/IEC 14496-3. Configuration data is mapped onto the octet string in an MSB-first basis. The first bit of the configuration data SHALL be located at the MSB of the first octet.

In the last octet, zero-padding bits, if necessary, shall follow the configuration data.

8.2.3 Indications for MPEG-4 video streams

8.2.3.1 Indication of profile and level

Senders MAY transmit the video profile and level indication in SDP.

The following syntax should be used:

a=fmtp:<format> profile-level-id=<value>

<value> being a decimal representation of MPEG-4 Visual Profile Level indication value (profile_and_level_indication) defined in Table G-1 of ISO/IEC 14496-2. This parameter MAY be used in the capability exchange or session setup procedure to indicate MPEG-4 Visual Profile and Level combination of which the MPEG-4 Visual codec is capable. If this parameter is not specified by the procedure, its default value of 1 (Simple Profile/Level 1) is used.

8.2.3.2 Indication of video decoder configuration

Senders MAY transmit the video decoder configuration in SDP. This parameter indicates the configuration of the corresponding MPEG-4 visual bitstream. It SHALL NOT be used to indicate the codec capability in the capability exchange procedure.

The following syntax should be used:

a=fmtp:<format> config=<value>

<value> being a hexadecimal representation of an octet string that expresses the MPEG-4 Visual configuration information, as defined in subclause 6.2.1 Start codes of ISO/IEC14496-2[2][4][9]. The configuration information is mapped onto the octet string in an MSB-first basis. The first bit of the configuration information SHALL be located at the MSB of the first octet. The configuration information indicated by this parameter SHALL be the same as the configuration information in the corresponding MPEG-4 Visual stream, except for first_half_vbv_occupancy and latter_half_vbv_occupancy, if it exists, which may vary in the repeated configuration information inside an MPEG-4 Visual stream (See 6.2.1 Start codes of ISO/IEC14496-2).

8.3 Concatenation of fmtp parameters

Multiple fmtp parameters SHOULD be expressed as a MIME media type string, in the form of a semicolon separated list of parameter=value pairs.

8.4 SDP file example

In the following is an example of SDP syntax for the description of a session containing one MPEG-4 audio stream, one MPEG-4 video and one MPEG-4 system stream, transported using this format. Note that the video stream Decoding Time Stamps are encoded on 4 bits in this example.

```
o= ....
I= ....
c=IN IP4 123.234.71.112
m=video 1034 RTP/AVT 97
a=fmtp:decodingtimeStampLength 4
a=rtpmap:97 mpeg4-sl
m=audio 810  RTP/AVT 98
a=rtpmpa:98 mpeg4-sl
m=application 1234  RTP/AVT 99
a=rtpmap:99 mpeg4-sl
```

9. Examples of usage of this payload format

9.1 MPEG-4 Video

Let us consider the case of a 30 frames per second MPEG-4 video stream which bit rate is high enough that Access Units have to be split in several SL packets (typically above 300 kb/s).

Let us assume also that the video codec generates in that case Video Packets suitable to fit in one SL packet i.e that the video codec is MTU aware and the MTU is 1500 bytes. We assume furthermore that this stream contains B frames and that decodingTimeStamps are present.

9.1.1 Typical SLConfigDescriptor for video streams

In this example the SLConfigDescriptor is:

```
class SLConfigDescriptor extends BaseDescriptor : bit(8)
tag=SLConfigDescrTag {
  bit(8) predefined;
```

```
    if (predefined==0) {
      bit(1) useAccessUnitStartFlag; = 1
      bit(1) useAccessUnitEndFlag; = 0
      bit(1) useRandomAccessPointFlag; = 1
      bit(1) hasRandomAccessUnitsOnlyFlag; = 0
      bit(1) usePaddingFlag; = 0
      bit(1) useTimeStampsFlag; = 1
      bit(1) useIdleFlag; = 0
      bit(1) durationFlag; = 0
      bit(32) timeStampResolution; = 30
      bit(32) OCRResolution; = 0
      bit(8) timeStampLength;  // must be <= 64  = 32
      bit(8) OCRLength;               // must be <= 64 = 0
      bit(8) AU_Length;               // must be <= 32 = 0
      bit(8) instantBitrateLength; = 0
      bit(4) degradationPriorityLength; = 0
```



```
      bit(5) AU_seqNumLength; // must be <= 16 = 0
      bit(5) packetSeqNumLength; // must be <= 16 = 0
      bit(2) reserved=0b11;
    }
    if (durationFlag) {
      bit(32) timeScale; // NOT USED
      bit(16) accessUnitDuration;  // NOT USED
      bit(16) compositionUnitDuration;  // NOT USED
    }
    if (!useTimeStampsFlag) {
      bit(timeStampLength) startDecodingTimeStamp; = 0
      bit(timeStampLength) startCompositionTimeStamp; = 0
    }
  }
```

Note that:
the useRandomAccessPointFlag is set so that the
randomAccessPointFlag can indicate that the corresponding SL packet
contains a GOV and the first Video Packet of an Intra coded frame.

9.1.2 Typical SL packet header structure for video streams

With this configuration we can extrapolate the following SL packet

```
header structure:
aligned(8) class SL_PacketHeader (SLConfigDescriptor SL) {
  if (SL.useAccessUnitStartFlag) bit(1) accessUnitStartFlag; // 1
bit
  if (accessUnitStartFlag) {
    if (SL.useRandomAccessPointFlag) bit(1) randomAccessPointFlag;
// 1 bit
    if (SL.useTimeStampsFlag) {
      bit(1) decodingTimeStampFlag; // 1 bit
      bit(1) compositionTimeStampFlag; // 1 bit
    }
    if (decodingTimeStampFlag) bit(SL.timeStampLength)
decodingTimeStamp;
    if (compositionTimeStampFlag) bit(SL.timeStampLength)
compositionTimeStamp;
  }
}
```

9.1.3 SDP mapping information

decodingTimeStamps are encoded on 32 bits, which is much more than needed for delta. Therefore the sender will use decodingTimeStampDeltaLength in the corresponding SDP to signal that only 6 bits are used for the coding of relative DTS in the RTP packet.

The remainingSLPacketHeaderSize cannot exceed 3 bits, which is encoded on 2 bits and signaled by remainingSLPacketHeaderSizeLength.

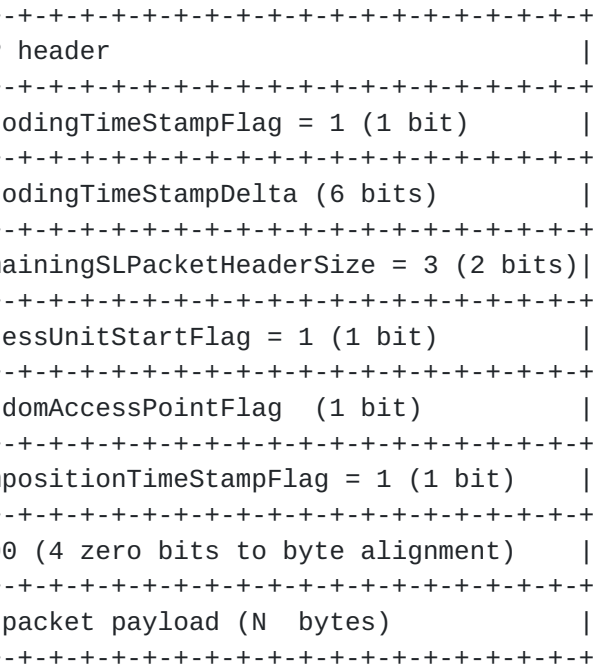
The resulting concatenated fmtp line is:


```
a=fmtp:<format> decodingTimeStampDeltaLength=6;
remainingSLPacketHeaderSizeLength=2
```

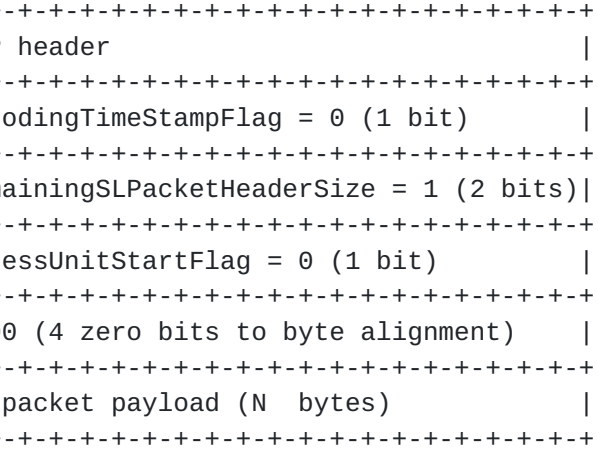
9.1.4 RTP packet structure

Such SL packet headers can result in several reduced SL packet headers:

For packets that transport first fragments of Access Units:



For packets that transport non-first fragments of Access Units:



Note the compositionTimeStamp is never present since it would be redundant with the RTP time stamp. However the value of compositionTimeStampFlag is still 1 to indicate that compositionTimeStamp was present for this SL packet and should therefore be restored by the receiver using the RTP time stamp.

In this example we have a RTP overhead of 40 + 2 bytes for 1400 bytes of payload i.e. 3 % overhead.

9.2 Low delay MPEG-4 Audio

This example is for a low delay service where a single SL packet is transported in each RTP packet.

9.2.1 Typical SLConfigDescriptor for low delay MPEG-4 Audio

Since CTS=DTS signaling of MPEG-4 time stamps is not needed.

We also assume here an audio Object Type for which all Access Units are
Random Access Points, which is signaled using the hasRandomAccessUnitsOnlyFlag in the SLConfigDescriptor.

In this example the SLConfigDescriptor is:

```
class SLConfigDescriptor extends BaseDescriptor : bit(8)
tag=SLConfigDescrTag {
  bit(8) predefined;
  if (predefined==0) {
    bit(1) useAccessUnitStartFlag; = 0
    bit(1) useAccessUnitEndFlag; = 0
    bit(1) useRandomAccessPointFlag; = 0
    bit(1) hasRandomAccessUnitsOnlyFlag; = 1
    bit(1) usePaddingFlag; = 0
    bit(1) useTimeStampsFlag; = 0
    bit(1) useIdleFlag; = 0
    bit(1) durationFlag; = 0
    bit(32) timeStampResolution; = 0
    bit(32) OCRResolution; = 0
    bit(8) timeStampLength;  // must be <= 64  = 0
    bit(8) OCRLength;                // must be <= 64 = 0
    bit(8) AU_Length;                // must be <= 32 = 0
    bit(8) instantBitrateLength; = 0
    bit(4) degradationPriorityLength; = 0
    bit(5) AU_seqNumLength; // must be <= 16 = 0
    bit(5) packetSeqNumLength; // must be <= 16 = 0
    bit(2) reserved=0b11;
  }
  if (durationFlag) {
    bit(32) timeScale; // NOT USED
    bit(16) accessUnitDuration;  // NOT USED
    bit(16) compositionUnitDuration;  // NOT USED
```

```
    }
    if (!useTimeStampsFlag) {
      bit(timeStampLength) startDecodingTimeStamp; = 0
      bit(timeStampLength) startCompositionTimeStamp; = 0
    }
  }
```


9.2.2 Typical SL packet header for low delay MPEG-4 Audio

With this configuration the SL header is empty.

This does not have to be indicated in SDP since the default value for remainingSLPacketHeaderSizeLength and decodingTimeStampDeltaLength is zero.
Therefore the absence of these fields in SDP indicates the absence of decodingTimeStampFlag and remainingSLPacketHeaderSize in RTP packets.

9.2.3 Overhead estimation for low delay MPEG-4 Audio

Depending on the actual MPEG-4 audio Object Type used the RTP overhead (IP+UDP+RTP headers) can be very large since the SL packet payload can be a few bytes or less.

9.3 Media delivery MPEG-4 Audio

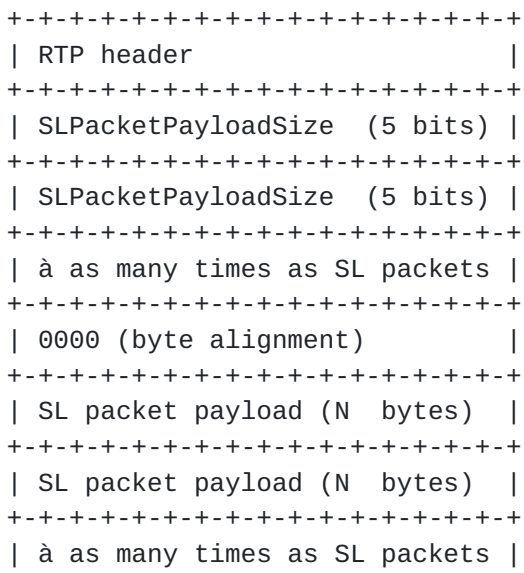
This example is for a service where delay is not an issue but streaming efficiency is of paramount importance. In this example multiple SL packets are transported in each RTP packet.

9.3.1 RTP packet structure

The SL configuration is the same as in the previous example; we will however use
SLPayloadSizeLength to indicate multiple SL packets per RTP packets. In this example we will assume that this size never exceeds 31 bytes and can therefore be encoded on 5 bits. This will be signaled in SDP using:

a=fmtp:<format> SLPayloadSizeLength=5

Therefore the structure of the RTP packet will be:



9.3.2 Overhead estimation for MPEG-4 Audio media delivery

The resulting overhead can be computed as follows:

At bit rate (BR) we compute the average Access Unit size (AvS) in bytes using the Access Unit duration (AuDur) in milliseconds as:
AvS = (int)(BR/8*AuDur/1000)

For example 8 kb/s CELP with AuDur=20 ms leads to AvS=20 bytes. In the same context as before we can assume 70 Access Units per RTP packets, therefore the overhead is 40 bytes for RTP+UDP+IP plus 70*5 bits = 44 bytes of SL headers i.e. the overhead is 6 %.

For high bit rate audio the number of SL packets per RTP packet will decrease, leading to better overhead figures.

9.4 Interleaving for MPEG-4 Audio

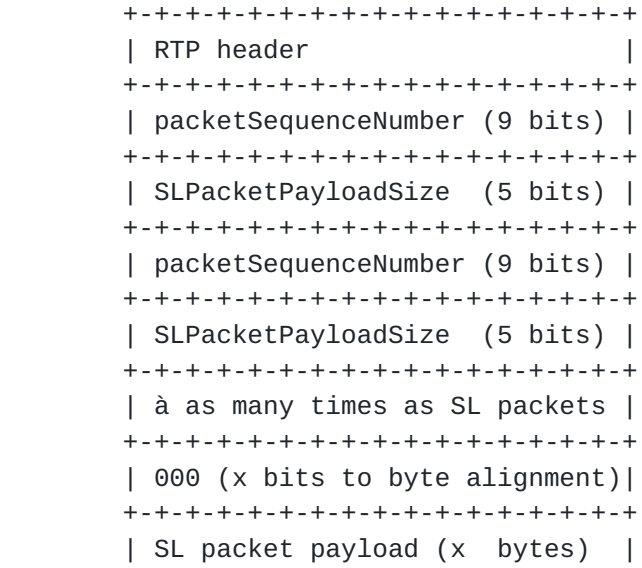
This example is the same as before with the addition of interleaving for error resilience.

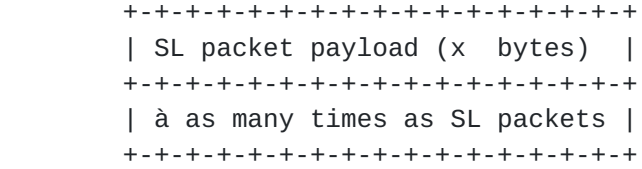
The SL configuration is the same as in the previous example except that packetSeqNumLength is not zero but 9 bits. We will also use SLPacketPayloadSizeLength to indicate multiple SL packets per RTP packets. Additionally we use packetSequenceNumberLength to signal the length
of all packetSequenceNumber fields (packetSequenceNumberDeltaLength is not used in this example)

This will be signaled in SDP using:

a=fmtp:<format> SLPayloadSizeLength=5;packetSequenceNumberLength=9

The RTP packet structure is then:





10. References

[1] ISO/IEC 14496-1:2000 MPEG-4 Systems October 2000

[2] ISO/IEC 14496-2:1999/Amd.1:2000(E) MPEG-4 Visual January 2000

[3] ISO/IEC 14496-3:1999/FDAM 1:20000 MPEG-4 Audio January 2000

[4] ISO/IEC 14496-6 FDIS Delivery Multimedia Integration Framework, November 1998.

[5] Schulzrinne, Casner, Frederick, Jacobson RTP: A Transport Protocol for Real Time Applications RFC 1889, Internet Engineering Task Force, January 1996.

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

[7] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, H. Kimata, RTP payload format for MPEG-4 Audio/Visual streams, work in progress, draft-ietf-avt-rtp-mpeg4-es-05.txt, September 2000.

[8] B. Thompson, T. Koren, D. Wing, Tunneling multiplexed Compressed RTP ("TCRTP"), work in progress, draft-ietf-avt-tcrtp-01.txt, July 2000.

[9] D. Singer, Y Lim, A Framework for the delivery of MPEG-4 over IP-based Protocols, work in progress, draft-singer-mpeg4-ip-01.txt,October 2000.

[10] Handley, Jacobson, SDP: Session Description Protocol, RFC 2327, Internet Engineering Task Force, April 1998.

11. Authors' Addresses

Olivier Avaro
France Telecom
35 A Sch tzenh ttenweg
60598 Frankfurt am Main
Deutschland
e-mail: olivier.avaro@francetelecom.fr

Andrea Basso
AT&T Labs Research
200 Laurel Avenue

Middletown, NJ 07748
USA
e-mail: basso@research.att.com

Stephen L. Casner
Packet Design, Inc.
66 Willow Place
Menlo Park, CA 94025
USA
casner@acm.org

M. Reha Civanlar
AT&T Labs - Research
100 Schultz Drive
Red Bank, NJ 07701
USA
e-mail: civanlar@research.att.com

Philippe Gentric
Philips Digital Networks
22 Avenue Descartes
94453 Limeil-Brevannes CEDEX
France
e-mail: philippe.gentric@philips.com

Carsten Herpel
THOMSON multimedia
Karl-Wiechert-Allee 74
30625 Hannover
Germany
e-mail: herpelc@thmulti.com

Zvi Lifshitz
Optibase Ltd.
7 Shenkar St.
Herzliya 46120
Israel
e-mail: zvil@optibase.com

Young-kwon Lim
mp4cast (MPEG-4 Internet Broadcasting Solution Consortium)
1001-1 Daechi-Dong Gangnam-Gu
Seoul, 305-333,
Korea

e-mail : young@techway.co.kr

Colin Perkins
USC Information Sciences Institute
4350 N. Fairfax Drive #620
Arlington, VA 22203
USA
e-mail : csp@isi.edu

Jan van der Meer
Philips Digital Networks
Cederlaan 4
5600 JB Eindhoven
Netherlands
e-mail : jan.vandermeer@philips.com