

draft-barbato-avt-rtp-theora-00
RTP Payload Format for Theora Encoded Video

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 28, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document describes a RTP payload format for transporting Theora encoded video. It details the RTP encapsulation mechanism for raw Theora data and configuration headers necessary to configure the decoder.

Also included within the document are the necessary details for the use of Theora with MIME and Session Description Protocol (SDP).

Editors Note

All references to RFC XXXX are to be replaced by references to the RFC number of this memo, when published.

Table of Contents

1. Introduction	3
1.1. Terminology	4
2. Payload Format	4
2.1. RTP Header	4
2.2. Payload Header	5
2.3. Payload Data	6
2.4. Example RTP Packet	7
3. Configuration Headers	8
3.1. In-band Header Transmission	9
3.1.1. Packed Configuration	9
3.2. Out of Band Transmission	10
3.2.1. Packed Headers	11
3.3. Loss of Configuration Headers	13
4. Comment Headers	13
5. Frame Packetizing	14
5.1. Example Fragmented Theora Packet	15
5.2. Packet Loss	17
6. IANA Considerations	18
6.1. Mapping MIME Parameters into SDP	19
6.1.1. SDP Example	20
6.2. Usage with the SDP Offer/Answer Model	20
7. Examples	21
7.1. Stream Video	21
8. Security Considerations	21
9. Acknowledgments	22
10. References	22
10.1. Normative References	22
10.2. Informative References	23
Author's Address	24
Intellectual Property and Copyright Statements	25

1. Introduction

Theora is a general purpose, lossy video codec. It is based on the VP3 video codec produced by On2 Technologies and has been donated to the Xiph.org Foundation.

Theora I is a block-based lossy transform codec that utilizes an 8 x 8 Type-II Discrete Cosine Transform and block-based motion compensation. This places it in the same class of codecs as MPEG-1, MPEG-2, MPEG-4, and H.263. The details of how individual blocks are organized and how DCT coefficients are stored in the bitstream differ substantially from these codecs, however. Theora supports only intra frames (I frames in MPEG) and inter frames (P frames in MPEG).

Theora provides none of its own framing, synchronization, or protection against transmission errors. Instead, the codec expects to receive a discrete sequence of data packets. Theora is a free-form variable bit rate (VBR) codec, and these packets have no minimum size, maximum size, or fixed/expected size. Theora packets are thus intended to be used with a transport mechanism that provides free-form framing, synchronization, positioning, and error correction in accordance with these design assumptions, such as Ogg [1] or RTP/AVP [3].

Theora I currently supports progressive video data of arbitrary dimensions at a constant frame rate in one of several Y'CbCr color spaces. Three different chroma subsampling formats are supported: 4:2:0, 4:2:2, and 4:4:4. The Theora I format does not support interlaced material, variable frame rates, bit-depths larger than 8 bits per component, nor alternate color spaces such as RGB or arbitrary multi-channel spaces. Black and white content can be efficiently encoded, however, because the uniform chroma planes compress well. Arbitrary frame size will be encoded rounding to the upper multiple of 16 both dimension for performance reason. The original width and height will be encoded in the header and the decoder will use this information to clip the decoded frame to the right dimensions.

Theora is similar to the Vorbis audio [10] in that it the decoder reads the probability model for the entropy coder and all quantization parameters from special "header" packets at the start of the compressed data. It is therefore impossible to decode any video data without having previously fetched the codec info and codec setup headers, although Theora can initiate decode at an arbitrary intra-frame packet so long as the codec has been initialized with the associated headers.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [2].

2. Payload Format

For RTP based transportation of Theora encoded video the standard RTP header is followed by a 4 octets payload header, then the payload data. The payload headers are used to associate the Theora data with its associated decoding codebooks as well as indicating if the following packet contains fragmented Theora data and/or the number of whole Theora data frames. The payload data contains the raw Theora bitstream information.

For RTP based transport of Theora encoded video the standard RTP header is followed by a 4 octets payload header, then the payload data.

2.1. RTP Header

The format of the RTP header is specified in [3] and shown in Figure 1. This payload format uses the fields of the header in a manner consistent with that specification.

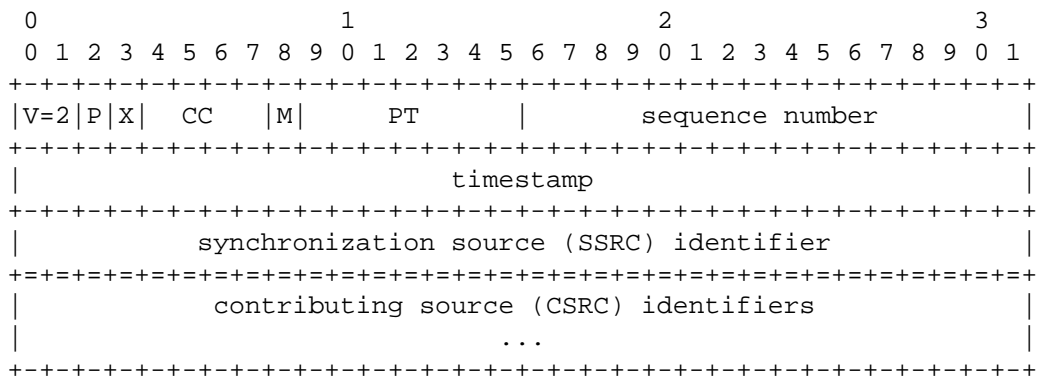


Figure 1: RTP Header

The RTP header begins with an octet of fields (V, P, X, and CC) to support specialized RTP uses (see [3] and [4] for details). For Theora RTP, the following values are used.

Version (V): 2 bits

This field identifies the version of RTP. The version used by this specification is two (2).

Padding (P): 1 bit

Padding MAY be used with this payload format according to section 5.1 of [3].

Extension (X): 1 bit

The Extension bit is used in accordance with [3].

CSRC count (CC): 4 bits

The CSRC count is used in accordance with [3].

Marker (M): 1 bit

The Marker bit is used in accordance with [3].

Payload Type (PT): 7 bits

An RTP profile for a class of applications is expected to assign a payload type for this format, or a dynamically allocated payload type SHOULD be chosen which designates the payload as Theora.

Sequence number: 16 bits

The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. This field is detailed further in [3].

Timestamp: 32 bits

A timestamp representing the presentation time of the first sample of the first Theora packet in the RTP packet. The clock frequency MUST be set to 90kHz.

SSRC/CSRC identifiers:

These two fields, 32 bits each with one SSRC field and a maximum of 16 CSRC fields, are as defined in [3].

2.2. Payload Header

The 4 octets following the RTP Header section are the Payload Header. This header is split into a number of bitfields detailing the format of the following Payload Data packets.

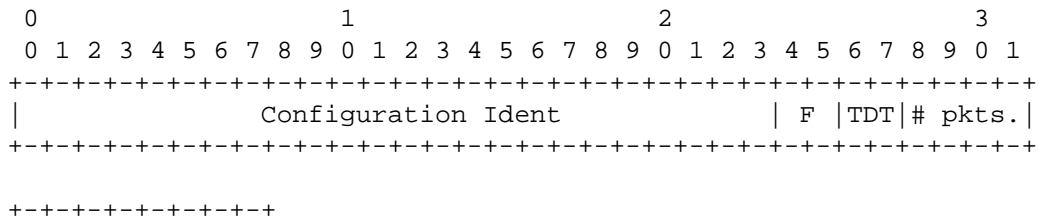


Figure 2: Payload Header

Configuration Ident: 24 bits

This 24 bit field is used to associate the Theora data to a decoding Packed Configuration.

Fragment type (F): 2 bit

This field is set according to the following list

- 0 = Not Fragmented
- 1 = Start Fragment
- 2 = Continuation Fragment
- 3 = End Fragment

This field must be zero if the number of packets field is non-zero.

Theora Data Type (TDT): 2 bits

This field sets the packet payload type for the Theora data. There are currently three Theora payload types.

- 0 = Raw Theora payload
- 1 = Theora Packed Configuration payload
- 2 = Legacy Theora Comment payload
- 3 = Reserved

The packets with a TDT of value 3 MUST be ignored

The last 4 bits represent the number of complete packets in this payload. This provides for a maximum number of 15 Theora packets in the payload. If the packet contains fragmented data the number of packets MUST be set to 0.

2.3. Payload Data

Each Theora payload section starts with a two octets length header that is used to represent the size of the following data payload, followed by the raw Theora packet data.

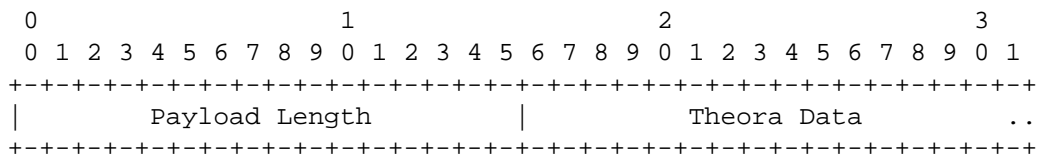


Figure 3: Payload Data

The Theora codec uses relatively unstructured raw packets containing binary integer fields of arbitrary width that often do not fall on an octet boundary. When a Theora encoder produces packets, unused space in the last byte of a packet is always zeroed during the encoding process. Thus, should this unused space be read, it will return binary zeros.

For payloads which consist of multiple Theora packets the payload data consists of the payload length field followed by the first Theora packet's data, then the payload length followed by the second Theora packet, and so on for each of the Theora packets in the payload.

2.4. Example RTP Packet

Here is an example RTP packet containing two Theora packets.

RTP Packet Header:

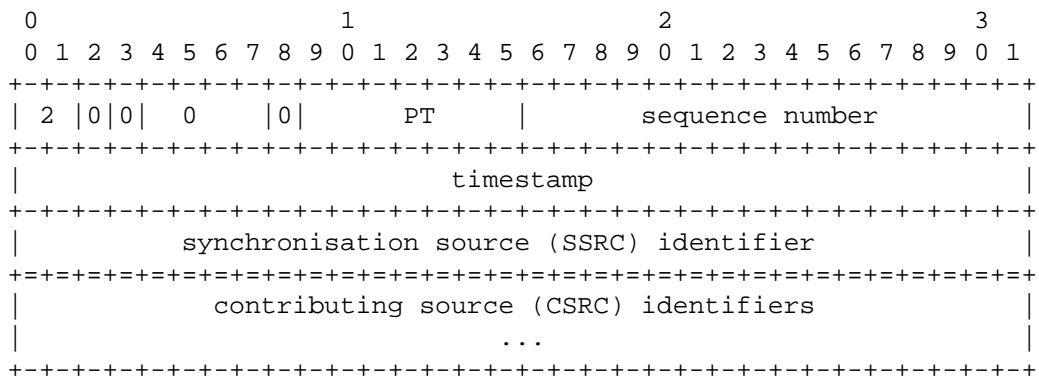


Figure 4: Example RTP Packet

Payload Data:

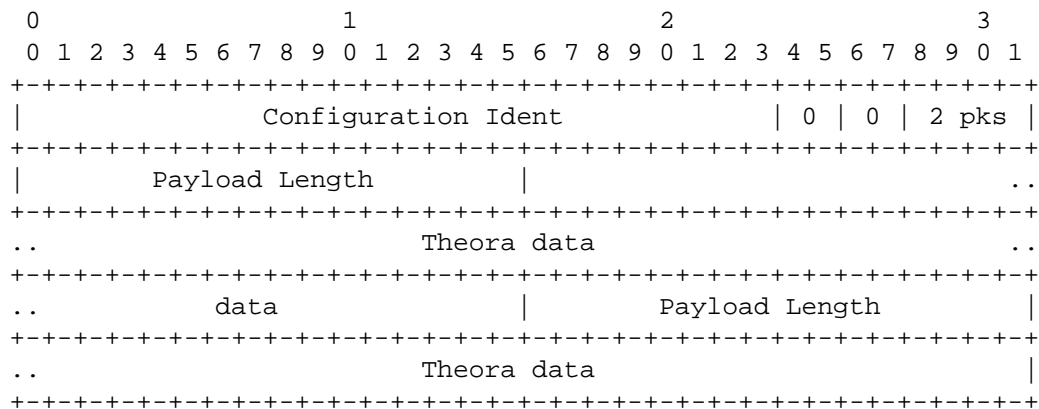


Figure 5: Example Theora Payload Packet

The payload portion of the packet begins with the 24 bit Configuration ident field followed by 8 bits describing the payload. The Fragment type field is set to 0, indicating that this packet contains whole Theora frame data. The Data type field is set to 0 since it is theora raw data. The number of whole Theora data packets is set to 2.

Each of the payload blocks starts with the two octets length field followed by the variable length Theora packet data.

3. Configuration Headers

To decode a Theora stream three configuration header packets are needed. The first, called the Identification Header, indicates the frame dimensions, quality, blocks used and the version of the Theora encoder used. The second, called the Comment Header, contains stream metadata and the third, called the Setup Header, contains details of the dequantization and Huffman tables.

Since this information must be transmitted reliably, and as the RTP stream may change certain configuration data mid-session, there are different methods for delivering this configuration data to a client, both in-band and out-of-band which are detailed below. SDP delivery is used to set up an initial state for the client application. The changes may be due to different dequantization and Huffman tables as well as different bitrates of the stream.

The delivery vectors in use are specified by an SDP attribute to indicate the method and the optional URI where the Vorbis Packed Configuration ([Section 3.1.1](#)) Packets could be fetched. Different delivery methods MAY be advertised for the same session. The in-band

codebook delivery SHOULD be considered as baseline, out-of-band delivery methods that don't use RTP will not be described in this document. For non chained streams, the Configuration delivery method RECOMMENDED is inline the Packed Configuration ([Section 3.1.1](#)) in the SDP as explained in the IANA considerations ([Section 6.1](#))

The 24 bit Ident field is used to map which Configuration will be used to decode a packet. When the Ident field changes, it indicates that a change in the stream has taken place. The client application MUST have in advance the correct configuration and if the client detects a change in the Ident value and does not have this information it MUST NOT decode the raw data associated until it fetches the correct Configuration.

3.1. In-band Header Transmission

The Packed Configuration ([Section 3.1.1](#)) Payload is sent in-band with the packet type bits set to match the payload type. Clients MUST be capable of dealing with periodic re-transmission of the configuration headers.

3.1.1. Packed Configuration

A Theora Packed Configuration is indicated with the payload type field set to 1. Of the three headers, defined in the Theora I specification [16], the identification and the setup will be packed together, the comment header is completely suppressed. It is up to the client to provide a minimal size comment header to the decoder if required by the implementation.

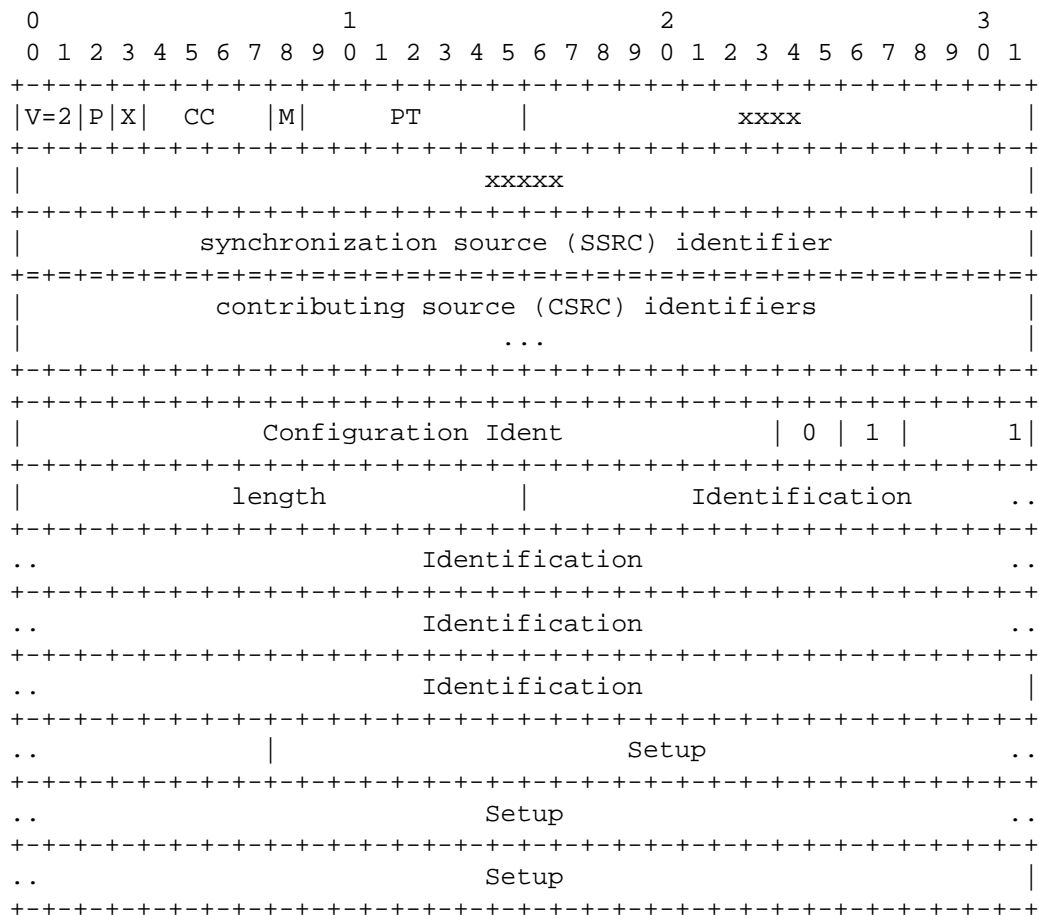


Figure 6: Packed Configuration Figure

The Ident field is set with the value that will be used by the Raw Payload Packets to address this Configuration. The Fragment type is set to 0 since the packet bears the full Packed configuration, the number of packet is set to 1. In practice, Packed Headers usually need to be fragmented to fit the path MTU.

3.2. Out of Band Transmission

This section, as stated above, does not cover all the possible out-of-band delivery methods since they rely on different protocols and be linked to specific applications. The following packet definition SHOULD be used in out-of-band delivery and MUST be used when Configuration is inlined in the SDP.

3.2.1. Packed Headers

As mentioned above, the recommended delivery vector for Theora configuration data is via a retrieval method that can be performed using a reliable transport protocol. As the RTP headers are not required for this method of delivery the structure of the configuration data is slightly different. The packed header starts with a 32 bit count field which details the number of packed headers that are contained in the bundle. Next is the Packed header payload for each setup id.

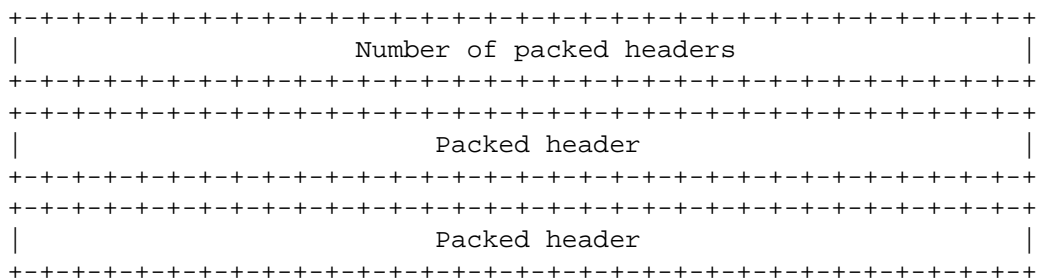


Figure 7: Packed Headers Overview

Since the Configuration Ident and the Identification Header are fixed length there is only a 16bit Length tag to define the length of the packed headers.

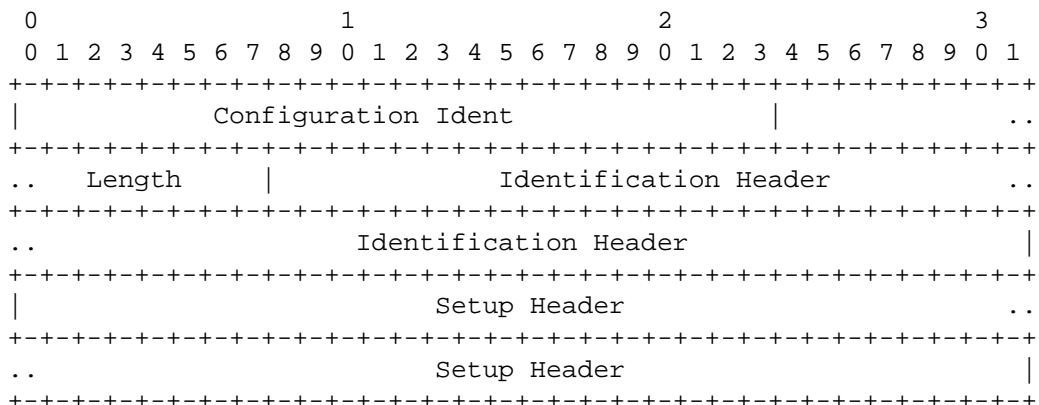


Figure 8: Packed Headers Detail

The key difference from the in-band format is that there is no need for the payload header octet.

3.2.1.1. Packed Headers IANA Considerations

The following IANA considerations MUST only be applied to the packed headers.

MIME media type name: audio

MIME subtype: theora-config

Required Parameters:

None

Optional Parameters:

None

Encoding considerations:

This media type contains binary data.

Security Considerations:

See [Section 6](#) of RFC XXXX.

Interoperability considerations:

None

Published specification:

RFC XXXX [RFC Editor: please replace by the RFC number of this memo, when published]

Applications which use this media type:

Theora encoded video, configuration data.

Additional information:

None

Person & email address to contact for further information:

Luca Barbato: <lu_zero@gentoo.org>
IETF Audio/Video Transport Working Group

Intended usage: COMMON

Restriction on usage:

This media type does not depend on the transport.

Author:

Luca Barbato

Change controller:

IETF AVT Working Group

3.3. Loss of Configuration Headers

Unlike the loss of raw Theora payload data, the loss of a configuration header can lead to a situation where it will not be possible to successfully decode the stream.

A loss of a Configuration Packet results in the halting of stream decoding and SHOULD be reported to the client as well as a loss report sent via RTCP.

4. Comment Headers

When the payload type is set to 2, the packet contains the comment metadata, such as artist name, track title and so on. These metadata messages are not intended to be fully descriptive but to offer basic title information. Clients MAY ignore them completely. The details on the format of the comments can be found in the Theora documentation [16].

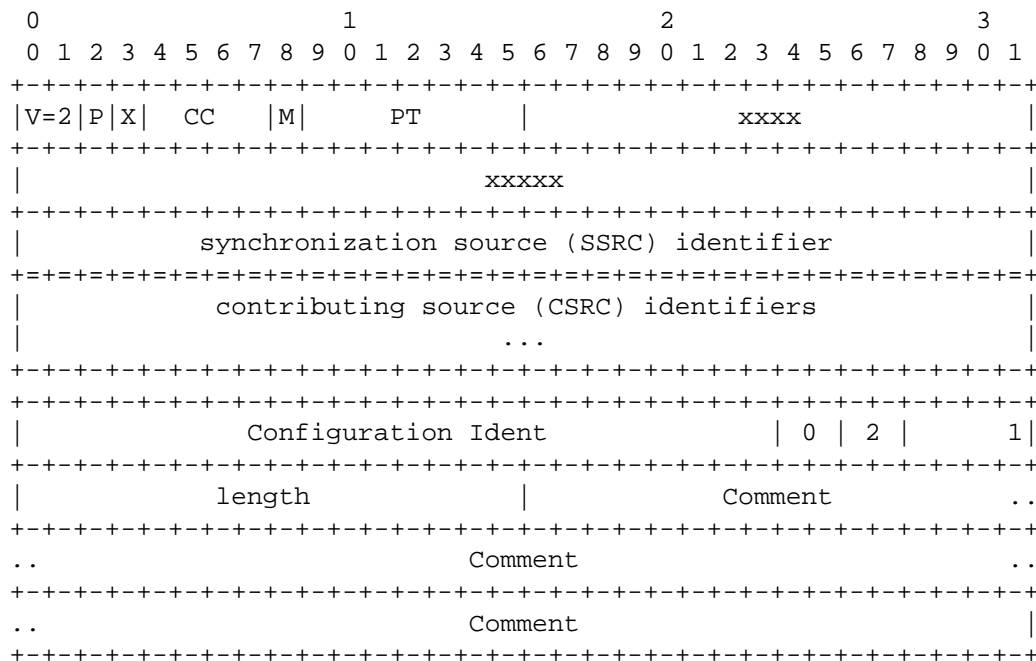


Figure 9: Comment Packet

The 2 byte length field is necessary since this Theora packet could be fragmented.

5. Frame Packetizing

Each RTP packet contains either one complete Theora packet, one Theora packet fragment, or an integer number of complete Theora packets (up to a maximum of 15 packets, since the number of packets is defined by a 4 bit value).

Any Theora data packet that is less than path MTU SHOULD be bundled in the RTP packet with as many Theora packets as will fit, up to a maximum of 15. Path MTU is detailed in [7] and [8].

A fragmented packet has a zero in the last four bits of the payload header. The RTP packet containing the first fragment will set the Fragment type to 1. Each RTP packet after the first will set the Fragment type to 2 in the payload header. The RTP packet containing the last fragment of the Theora packet will have the Fragment type set to 3. If the fragmented Theora packet spans only two RTP packets, the first will set the Fragment type field to 1 and the second will set it to 2. To maintain the correct sequence for fragmented packet reception the timestamp field of fragmented packets

MUST be the same as the first packet sent, with the sequence number incremented as normal for the subsequent RTP packets.

5.1. Example Fragmented Theora Packet

Here is an example fragmented Theora packet split over three RTP packets. Each packet contains the standard RTP headers as well as the 4 octets Theora headers.

Packet 1:

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| V=2 | P | X | CC | M | PT |           1000           |
+-----+-----+-----+-----+-----+-----+-----+
|                                     xxxxxx              |
+-----+-----+-----+-----+-----+-----+-----+
|               synchronization source (SSRC) identifier |
+=====+=====+=====+=====+=====+=====+=====+
|               contributing source (CSRC) identifiers    |
|                                     ...                  |
+-----+-----+-----+-----+-----+-----+-----+
|               Configuration Ident           | 1 | 0 | 0 |
+-----+-----+-----+-----+-----+-----+-----+
|               Payload Length           | Theora data    |..
+-----+-----+-----+-----+-----+-----+-----+
|..                                     Theora data       |..
+-----+-----+-----+-----+-----+-----+-----+

```

Figure 10: Example Fragmented Packet (Packet 1)

In this packet the initial sequence number is 1000 and the timestamp is xxxxxx. The Fragment type field is set to one, indicating it is the start packet of a serie of fragments. The number of packets field is set to 0, and as the payload is raw Theora data the Theora payload type field is set to 0.

Packet 2:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|V=2|P|X|  CC  |M|      PT      |              1001              |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     xxxxx                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               synchronization source (SSRC) identifier           |
+=====+=====+=====+=====+=====+=====+=====+=====+
|               contributing source (CSRC) identifiers               |
|                                     ...                             |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Configuration Ident          | 2 | 0 |          0 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Payload Length              | ..                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
..                                     Theora data                    ..
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 11: Example Fragmented Packet (Packet 2)

The Fragment type field is set to 2 and the number of packets field is set to 0. For large Theora fragments there can be several of these type of payload packets. The maximum RTP packet size SHOULD be no greater than the path MTU, including all RTP and payload headers. The sequence number has been incremented by one but the timestamp field remains the same as the initial packet.

Packet 3:

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|V=2|P|X|  CC   |M|      PT      |              1002              |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     xxxxx                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               synchronization source (SSRC) identifier           |
+=====+=====+=====+=====+=====+=====+=====+=====+
|               contributing source (CSRC) identifiers               |
|                                     ...                             |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Configuration Ident                                | 3 | 0 |      0 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Payload Length                                |      ..
+-----+-----+-----+-----+-----+-----+-----+-----+
..                                     Theora data                                     ..
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 12: Example Fragmented Packet (Packet 3)

This is the last Theora fragment packet. The Fragment type field is set to 3 and the packet count remains set to 0. As in the previous packets the timestamp remains set to the first packet in the sequence and the sequence number has been incremented.

5.2. Packet Loss

As there is no error correction within the Theora stream, packet loss will result in a loss of signal. Packet loss is more of an issue for fragmented Theora packets as the client will have to cope with the handling of the Fragment type field. If we use the fragmented Theora packet example above and the first packet is lost the client **MUST** detect that the next packet has the packet count field set to 0 and the Fragment type is set to 2 and **MUST** drop it. The next packet, which is the final fragmented packet, **MUST** be dropped in the same manner. Feedback reports on lost and dropped packets **MUST** be sent back via RTCP.[note: reordering]

If a particular multicast session has a large number of participants care must be taken to prevent an RTCP feedback implosion, [9], in the event of packet loss from a large number of participants.

Loss of any of the Configuration fragment will result in the loss of the full Configuration packet as detailed in the Loss of

Configuration Headers ([Section 3.3](#)) section.

6. IANA Considerations

MIME media type name: video

MIME subtype: theora

Required Parameters:

sampling: Determines the chroma subsampling format.

width: Determines the number of pixels per line. This is an integer between 1 and 1048561 and MUST be in multiples of 16.

height: Determines the number of lines per frame encoded. This is an integer between 1 and 1048561 and MUST be in multiples of 16.

delivery-method: indicates the delivery methods in use, the possible values are: inline, in_band, out_band/specific_name Where "specific_name" is the name of the out of band delivery method.

configuration: the base16 [\[11\]](#) (hexadecimal) representation of the Packed Headers ([Section 3.2.1](#)).

Optional Parameters:

configuration-uri: the URI of the configuration headers in case of out of band transmission. In the form of "protocol://path/to/resource/". Depending on the specific method the single ident packets could be retrieved by their number or aggregated in a single stream, aggregates MAY be compressed using gzip [\[12\]](#) or bzip2 [\[14\]](#) and an sha1 [\[13\]](#) checksum MAY be provided in the form of "protocol://path/to/resource/aggregated.bz2!shalhash"

Encoding considerations:

This media type is framed and contains binary data.

Security Considerations:

See [Section 6](#) of RFC XXXX.

Interoperability considerations:

None

Published specification:

RFC XXXX [RFC Editor: please replace by the RFC number of this memo, when published]

Ogg Theora I specification: Codec setup and packet decode.
Available from the Xiph website, <http://www.xiph.org>

Applications which use this media type:

Audio streaming and conferencing tools

Additional information:

None

Person & email address to contact for further information:

Luca Barbato: <lu_zero@gentoo.org>
IETF Audio/Video Transport Working Group

Intended usage:

COMMON

Restriction on usage:

This media type depends on RTP framing, and hence is only defined for transfer via RTP [3]

Author:

Luca Barbato

Change controller:

IETF AVT Working Group

6.1. Mapping MIME Parameters into SDP

The information carried in the MIME media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [6], which is commonly used to describe RTP sessions. When SDP is

used to specify sessions the mapping are as follows:

- o The MIME type ("video") goes in SDP "m=" as the media name.
- o The MIME subtype ("theora") goes in SDP "a=rtpmap" as the encoding name.
- o The clock rate in the "a=rtpmap" line MUST be 90000
- o The mandated parameters "delivery-method" and "configuration" MUST be included in the SDP "a=fmpt" attribute.
- o The optional parameter "configuration-uri", when present, MUST be included in the SDP "a=fmpt" attribute and MUST follow the delivery-method that applies.

If the stream uses multiple decoder setup configurations and all of them are known in advance, the Configuration Packet for each file SHOULD be packaged together and passed to the client using the configuration attribute.

The URI specified in the configuration-uri attribute MUST point to a location where all of the Configuration Packets needed for the life of the session reside.

6.1.1. SDP Example

The following example shows a basic SDP for a single stream. The first configuration packet is inlined in the sdp, other configurations could be fetched at any time from the first provided uri using or all the known configuration could be downloaded using the second uri. The inline base16 [11] configuration string is omitted because of the length.

```
c=IN IP4 192.0.0.1
m=video RTP/AVP 98
a=rtpmap:98 theora/90000
a=fmtp:98 sampling=YCbCr-4:2:2; width=1280; height=720; delivery-
method=inline; configuration=base16string1; delivery-
method=out_band/rtsp; delivery-method=out_band/rtsp;
configuration-uri=rtsp://path/to/resource/; delivery-
method=out_band/http; configuration-uri=http://another/path/to/
resource/aggregate.bz2!shash;
```

6.2. Usage with the SDP Offer/Answer Model

The offer, as described in An Offer/Answer Model Session Description Protocol [5], may contain a large number of delivery methods per single fmtp attribute, the answerer MUST remove every delivery-method

and configuration-uri not supported. All the parameters MUST not be altered on answer otherwise.

7. Examples

The following examples are common usage patterns that MAY be applied in such situations, the main scope of this section is to explain better usage of the transmission vectors.

7.1. Stream Video

This is one of the most common situation: one single server streaming content in multicast, the clients may start a session at random time. The content itself could be a mix of live stream, as the wj's voice or studio scenes, and stored streams, as the music she plays.

In this situation we don't know in advance how many codebooks we will use. The clients can join anytime and users expect to start the fruition of the content in a short time.

On join the client will receive the current Configuration necessary to decode the current streams inlined in the SDP so that the decoding will start immediately after.

When the streamed content changes the new Configuration is sent in-band before the actual stream, and the Configuration that has to be sent inline in the SDP updated. Since the inline method is unreliable, an out of band fallback is provided.

The client could choose to fetch the Configuration from the alternate source as soon it discovers a Configuration packet got lost inline or use selective retransmission [17], if the server supports the feature.

A serverside optimization would be to keep an hash list of the Configurations per session to avoid packing all of them and send the same Configuration with different Ident tags

A clientside optimization would be to keep a tag list of the Configurations per session and don't process configuration packets already known.

8. Security Considerations

RTP packets using this payload format are subject to the security considerations discussed in the RTP specification [3]. This implies

that the confidentiality of the media stream is achieved by using encryption. Because the data compression used with this payload format is applied end-to-end, encryption may be performed on the compressed data. Where the size of a data block is set care MUST be taken to prevent buffer overflows in the client applications.

9. Acknowledgments

This document is a continuation of [draft-kerr-avt-theora-rtp-00.txt](#)

Thanks to the AVT, Ogg Theora Communities / Xiph.org, Fluendo, Ralph Giles, Mike Smith, Phil Kerr, Timothy Terriberry, Stefan Ehmann, Alessandro Salvatori, Politecnico di Torino (LS)^3/IMG Group in particular Federico Ridolfo, Francesco Varano, Giampaolo Mancini, Juan Carlos De Martin.

10. References

10.1. Normative References

- [1] Pfeiffer, S., "The Ogg Encapsulation Format Version 0", [RFC 3533](#).
- [2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#).
- [3] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for real-time applications", [RFC 3550](#).
- [4] Schulzrinne, H. and S. Casner, "RTP Profile for video and Video Conferences with Minimal Control.", [RFC 3551](#).
- [5] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", [RFC 3264](#).
- [6] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", [RFC 2327](#).
- [7] Mogul et al., J., "Path MTU Discovery", [RFC 1063](#).
- [8] McCann et al., J., "Path MTU Discovery for IP version 6", [RFC 1981](#).
- [9] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for RTCP-based Feedback (RTP/AVPF)",

Internet Draft ([draft-ietf-avt-rtcp-feedback-11](#): Work in progress).

- [10] Barbato, L., "RTP Payload Format for Vorbis Encoded Audio - [draft-ietf-avt-vorbis-rtp-00](#)", Internet Draft (Work in progress).
- [11] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 3548](#).
- [12] Deutsch, P., "GZIP file format specification version 4.3", [RFC 1952](#).
- [13] National Institute of Standards and Technology, "Secure Hash Standard", May 1993.
- [14] Seward, J., "libbz2 and bzip2".

10.2. Informative References

- [15] "libTheora: Available from the Xiph website, <http://www.xiph.org>".
- [16] "Theora I specification: Codec setup and packet decode. http://www.xiph.org/theora/doc/Theora_I_spec.pdf".
- [17] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", [RFC 3611](#), November 2003.
- [18] "ITU-T Recommendation V.42, 1994, Rev. 1. Error-correcting Procedures for DCEs Using Asynchronous-to-Synchronous Conversion. International Telecommunications Union. Available from the ITU website, <http://www.itu.int>".
- [19] "ISO 3309, October 1984, 3rd Edition. Information Processing Systems--Data Communication High-Level Data Link Control Procedure--Frame Structure. International Organization for Standardization."

Author's Address

Luca Barbato
Xiph.Org

Email: lu_zero@gentoo.org
URI: <http://www.xiph.org/>

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.