AVT Working Group
Internet-Draft
Expires: August 20, 2008

L. Barbato
Xiph

Feb 17, 2008

RTP Payload Format for Vorbis Encoded Audio
draft-ietf-avt-rtp-vorbis-09

Status of This Memo

Copyright Notice

Abstract

This document describes an RTP payload format for transporting Vorbis encoded audio. It details the RTP encapsulation mechanism for raw Vorbis data and details the delivery mechanisms for the decoder probability model, referred to as a codebook and other setup information.

Also included within this memo are media type registrations, and the details necessary for the use of Vorbis with the 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

Vorbis is a general purpose perceptual audio codec intended to allow maximum encoder flexibility, thus allowing it to scale competitively over an exceptionally wide range of bitrates. At the high quality/ bitrate end of the scale (CD or DAT rate stereo, 16/24 bits), it is in the same league as MPEG-4 AAC. Vorbis is also intended for lower and higher sample rates (from 8kHz telephony to 192kHz digital masters) and a range of channel representations (monaural, polyphonic, stereo, quadraphonic, 5.1, ambisonic, or up to 255 discrete channels).

Vorbis encoded audio is generally encapsulated within an Ogg format bitstream [11], which provides framing and synchronization. For the purposes of RTP transport, this layer is unnecessary, and so raw Vorbis packets are used in the payload.

1.1. Conformance and Document Conventions

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 BCP 14, [1] and indicate requirement levels for compliant implementations. Requirements apply to all implementations unless otherwise stated.

An implementation is a software module that supports one of the media types defined in this document. Software modules may support multiple media types, but conformance is considered individually for each type.

Implementations that fail to satisfy one or more "MUST" requirements are considered non-compliant. Implementations that satisfy all "MUST" requirements, but fail to satisfy one or more "SHOULD" requirements, are said to be "conditionally compliant". All other implementations are "unconditionally compliant".

2. Payload Format

For RTP based transport of Vorbis encoded audio the standard RTP header is followed by a 4 octets payload header, then the payload data. The payload headers are used to associate the Vorbis data with its associated decoding codebooks as well as indicating if the following packet contains fragmented Vorbis data and/or the number of whole Vorbis data frames. The payload data contains the raw Vorbis bitstream information. There are 3 types of Vorbis data, an RTP payload MUST contain just one of them at a time.

2.1. RTP Header

The format of the RTP header is specified in [2] and shown in Figure Figure 1. This payload format uses the fields of the header in a manner consistent with that specification.

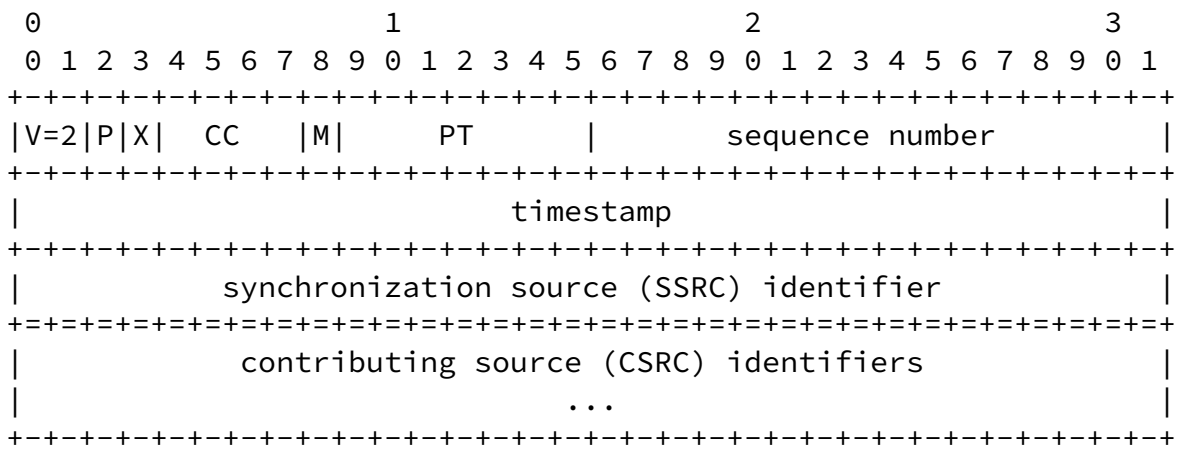


Figure 1: RTP Header

The RTP header begins with an octet of fields (V, P, X, and CC) to support specialized RTP uses (see [2] and [3] for details). For Vorbis 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 [2].

Extension (X): 1 bit

The Extension bit is used in accordance with [2].

CSRC count (CC): 4 bits

The CSRC count is used in accordance with [2].

Marker (M): 1 bit

Set to zero. Audio silence suppression not used. This conforms to section 4.1 of [10].

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 Vorbis.

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 [2].

Timestamp: 32 bits

A timestamp representing the sampling time of the first sample of the first Vorbis packet in the RTP payload. The clock frequency MUST be set to the sample rate of the encoded audio data and is conveyed out-of-band (e.g. as an SDP parameter).

SSRC/CSRC identifiers:

These two fields, 32 bits each with one SSRC field and a maximum of

16 CSRC fields, are as defined in [2].

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.

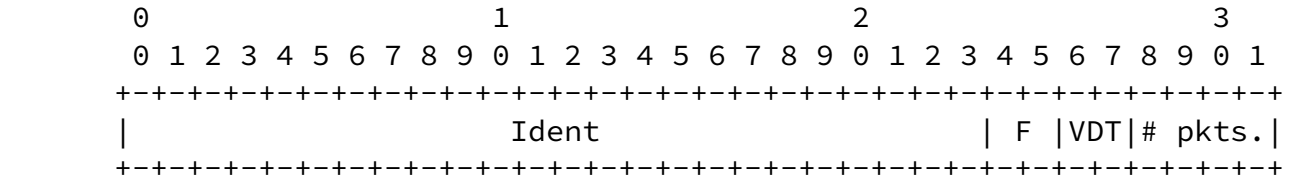


Figure 2: Payload Header

Ident: 24 bits

This 24 bit field is used to associate the Vorbis data to a decoding Configuration. It is stored as network byte order integer.

Fragment type (F): 2 bits

This field is set according to the following list

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

Vorbis Data Type (VDT): 2 bits

This field specifies the kind of Vorbis data stored in this RTP packet. There are currently three different types of Vorbis payloads. Each packet MUST contain only a single type of Vorbis packet (e.g. you must not aggregate configuration and comment packets in the same RTP payload)

0 = Raw Vorbis payload
1 = Vorbis Packed Configuration payload

2 = Legacy Vorbis Comment payload
3 = Reserved

The packets with a VDT 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 Vorbis packets in the payload. If the payload contains fragmented data the number of packets MUST be set to 0.

2.3. Payload Data

Raw Vorbis packets are currently unbounded in length, application profiles will likely define a practical limit. Typical Vorbis packet sizes range from very small (2-3 bytes) to quite large (8-12 kilobytes). The reference implementation [12] typically produces packets less than ~800 bytes, except for the setup header packets which are ~4-12 kilobytes. Within an RTP context, to avoid fragmentation, the Vorbis data packet size SHOULD be kept sufficiently small so that after adding the RTP and payload headers, the complete RTP packet is smaller than the path MTU.

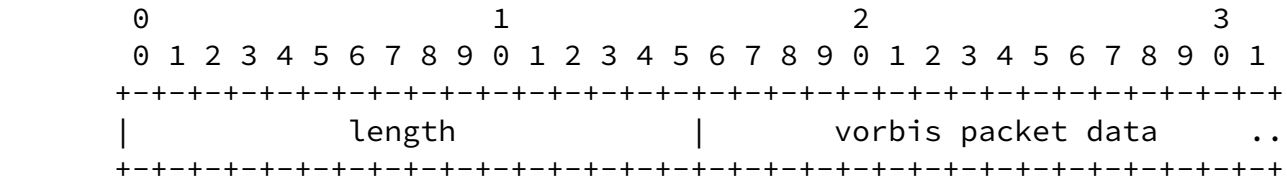


Figure 3: Payload Data Header

Each Vorbis payload packet starts with a two octet length header, which is used to represent the size in bytes of the following data payload, followed by the raw Vorbis data padded to the nearest byte

boundary, as explained by the vorbis specification [10]. The length value is stored as network byte order integer.

For payloads which consist of multiple Vorbis packets the payload data consists of the packet length followed by the packet data for each of the Vorbis packets in the payload.

The Vorbis packet length header is the length of the Vorbis data

block only and does not include the length field.

The payload packing of the Vorbis data packets MUST follow the guidelines set-out in [3] where the oldest Vorbis packet occurs immediately after the RTP packet header. Subsequent Vorbis packets, if any, MUST follow in temporal order.

Channel mapping of the audio is in accordance with the Vorbis I Specification [10].

2.4. Example RTP Packet

Here is an example RTP payload containing two Vorbis packets.

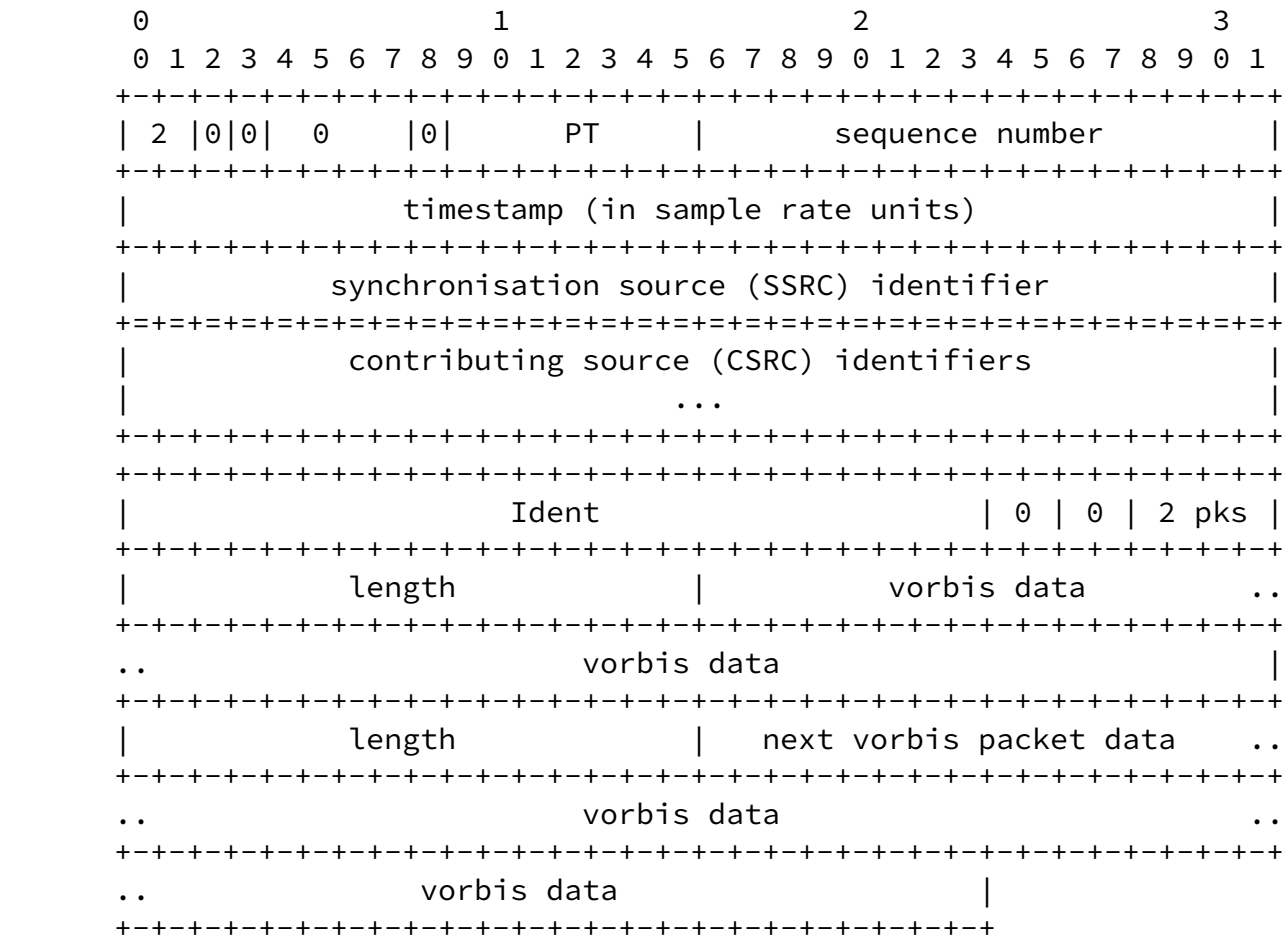


Figure 4: Example Raw Vorbis Packet

The payload data section of the RTP packet begins with the 24 bit Ident field followed by the one octet bitfield header, which has the number of Vorbis frames set to 2. Each of the Vorbis data frames is prefixed by the two octets length field. The Packet Type and Fragment Type are set to 0. The Configuration that will be used to decode the packets is the one indexed by the ident value.

3. Configuration Headers

Unlike other mainstream audio codecs Vorbis has no statically configured probability model. Instead, it packs all entropy decoding configuration, Vector Quantization and Huffman models into a data block that must be transmitted to the decoder along with the compressed data. A decoder also requires information detailing the number of audio channels, bitrates and similar information to configure itself for a particular compressed data stream. These two blocks of information are often referred to collectively as the "codebooks" for a Vorbis stream, and are nominally included as special "header" packets at the start of the compressed data. In addition, the Vorbis I specification [10] requires the presence of a comment header packet which gives simple metadata about the stream, but this information is not required for decoding the frame sequence.

Thus these two codebook header packets must be received by the decoder before any audio data can be interpreted. These requirements pose problems in RTP, which is often used over unreliable transports.

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 is detailed below. In order to set up an initial state for the client application the configuration MUST be conveyed via the signalling channel used to setup the session. One example of such signalling is SDP [5] with the Offer/ Answer Model [8]. Changes to the configuration MAY be communicated via a re-invite, conveying new SDP, or sent in-band in the RTP channel. Implementations MUST support in-band delivery of updated codebooks, and SHOULD support out-of-band codebook update using a new SDP file. The changes may be due to different codebooks as well as different bitrates of the RTP stream.

For non chained streams, the recommended Configuration delivery method is inline the Packed Configuration (Section 3.1.1) in the SDP as explained in the IANA considerations (Section 7.1).

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 Vorbis 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 Vorbis Data Type. Clients MUST be capable of dealing with fragmentation and periodic re-transmission of [14] the configuration headers. The RTP timestamp value MUST reflect the transmission time of the first data packet for which this configuration applies.

3.1.1. Packed Configuration

A Vorbis Packed Configuration is indicated with the Vorbis Data Type field set to 1. Of the three headers defined in the Vorbis I specification [10], the Identification and the Setup MUST be packed as they are, while the comment header MAY be replaced with a dummy one.

The packed configuration follows a generic way to store Xiph codec configurations: The first field stores the number of the following packets minus one (count field), the next ones represent the size of the headers (length fields), the headers immediately follow the list of length fields. The size of the last header is implicit.

The count and the length fields are encoded using the following logic: the data is in network byte order, every byte has the most significant bit used as flag and the following 7 used to store the value. The first N bit are to be taken, where N is number of bits needed to represent the value, taken modulo 7, and stored in the first byte. If there are more bits, the flag bit is set to 1 and the subsequent 7bit are stored in the following byte, if there are remaining bits set the flag to 1 and the same procedure is repeated. The ending byte has the flag bit set to 0. In order to decode it is enough to iterate over the bytes until the flag bit set to 0, for every byte the data is added to the accumulated value multiplied by 128.

The headers are packed in the same order they are present in ogg: Identification, Comment, Setup.

The 2 byte length tag defines the length of the packed headers as the sum of the Configuration, Comment and Setup lengths.

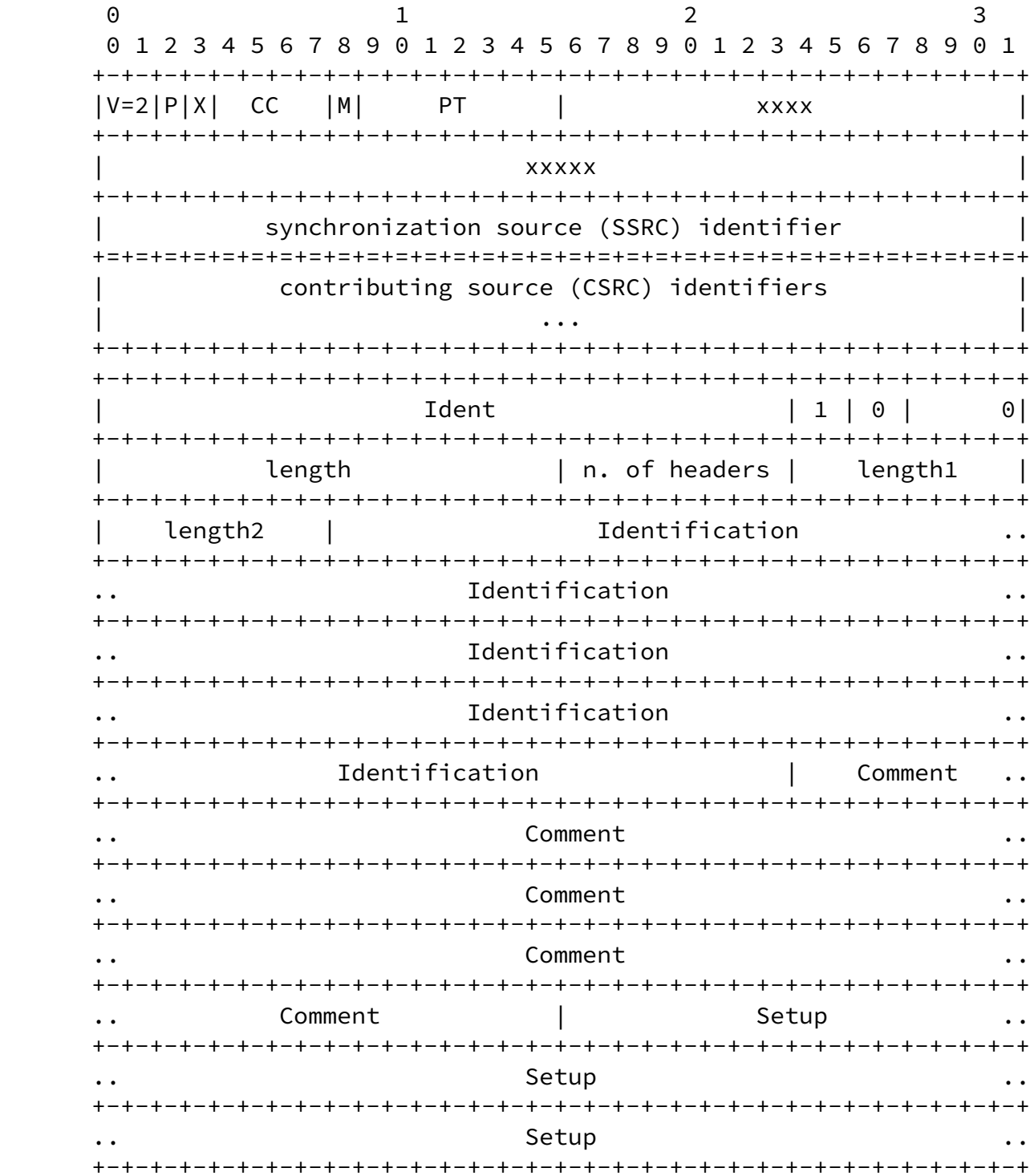


Figure 5: Packed Configuration Figure

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.

3.2. Out of Band Transmission

The following packet definition MUST be used when Configuration is inlined in the SDP.

3.2.1. Packed Headers

As mentioned above the RECOMMENDED delivery vector for Vorbis 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 (network byte ordered) count field which details the number of packed headers that are contained in the bundle. Next is the Packed header payload for each chained Vorbis stream.

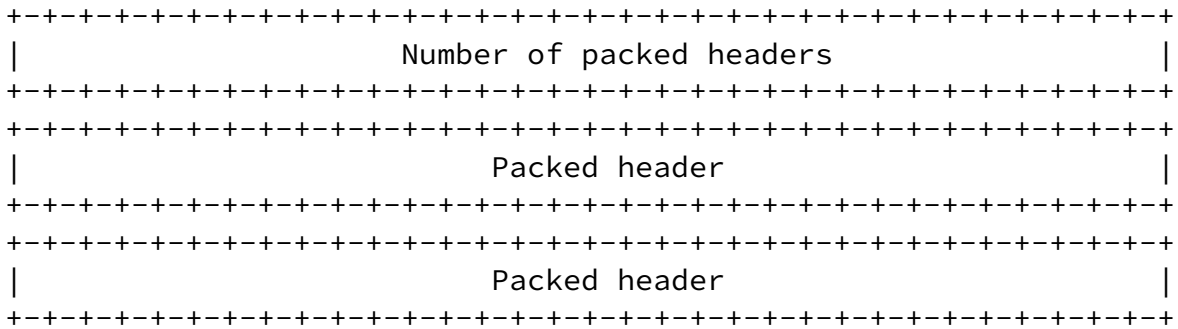


Figure 6: Packed Headers Overview

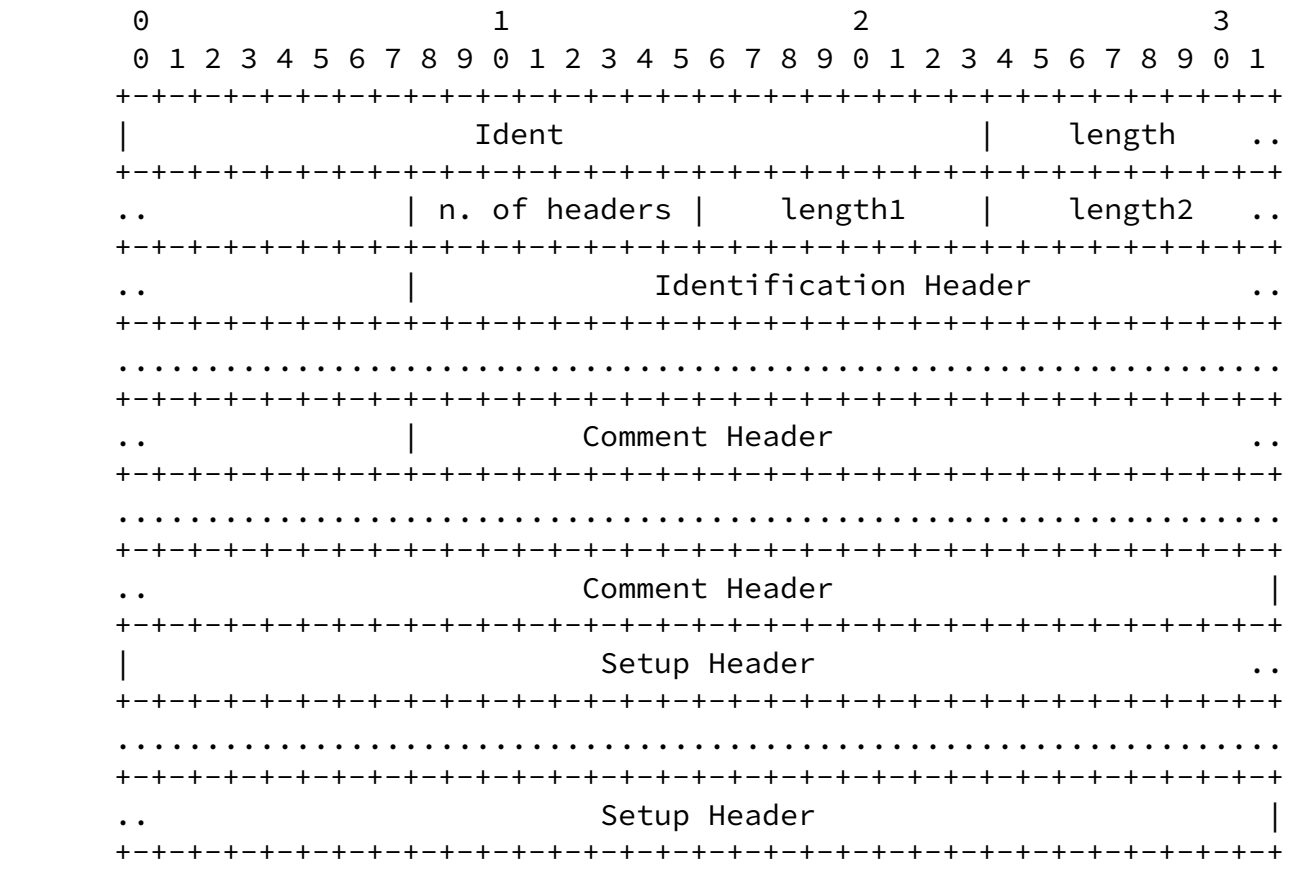


Figure 7: Packed Headers Detail

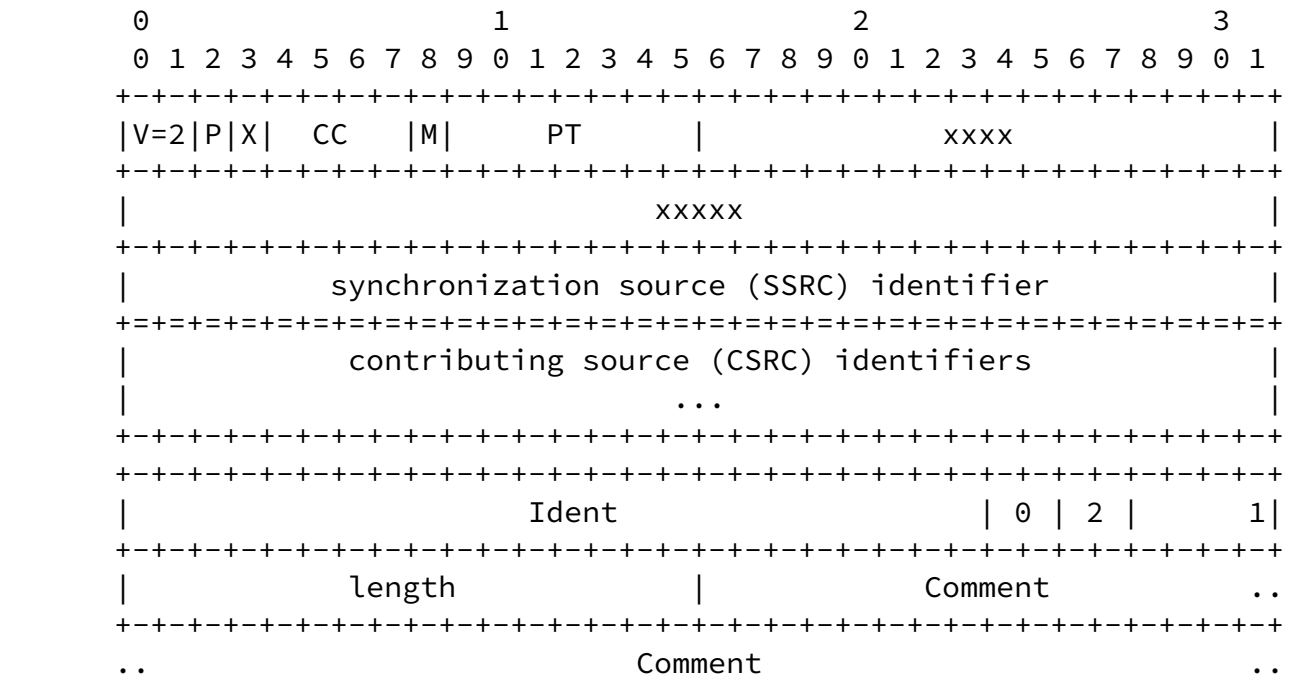
The key difference between the in-band format and this one, is that there is no need for the payload header octet. In this figure the comment has a size bigger than 127 bytes.

3.3. Loss of Configuration Headers

Unlike the loss of raw Vorbis payload data, loss of a configuration header lead to a situation where it will not be possible to successfully decode the stream. Implementations MAY try to recover from error by requesting again the missing Configuration or, if the delivery method is in-band, by buffering the payloads waiting for the Configuration needed to decode them. The baseline reaction SHOULD be either reset or end the RTP session.

4. Comment Headers

With the Vorbis Data Type flag set to 2, this indicates that the packet contain 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 track/song information. Clients MAY ignore it completely. The details on the format of the comments can be found in the Vorbis documentation [10].



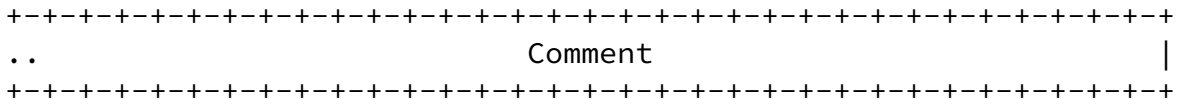


Figure 8: Comment Packet

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

5. Frame Packetization

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

Any Vorbis data packet that is less than path MTU SHOULD be bundled in the RTP payload with as many Vorbis packets as will fit, up to a maximum of 15, except when such bundling would exceed an application's desired transmission latency. Path MTU is detailed in [6] and [7].

A fragmented packet has a zero in the last four bits of the payload header. The first fragment will set the Fragment type to 1. Each fragment after the first will set the Fragment type to 2 in the payload header. The consecutive fragments MUST be sent without any other payloads being sent between the first and the last fragment. The RTP payload containing the last fragment of the Vorbis packet will have the Fragment type set to 3. 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 payloads, this will affect the RTCP jitter measurement. The length field shows the fragment length.

5.1. Example Fragmented Vorbis Packet

Here is an example fragmented Vorbis packet split over three RTP payloads. Each of them contains the standard RTP headers as well as the 4 octets Vorbis headers.

Packet 1:

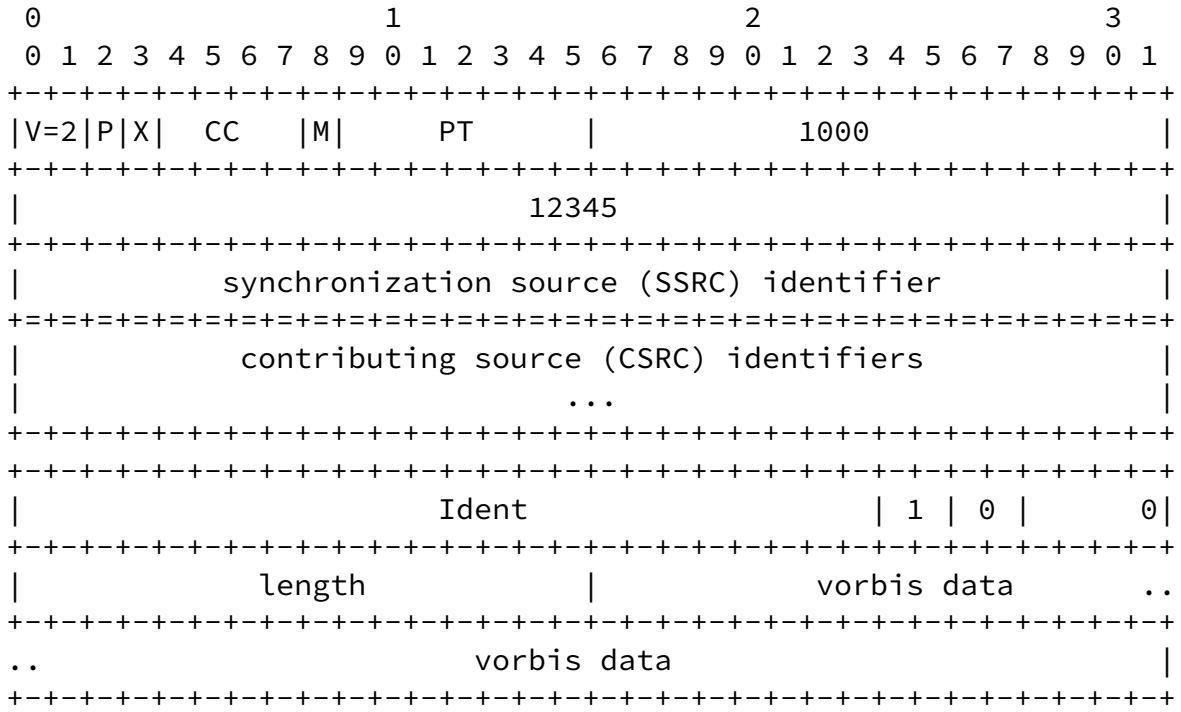
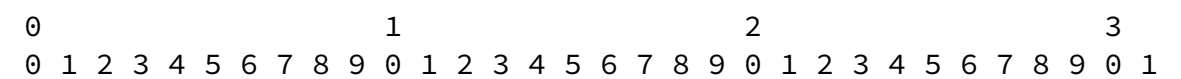


Figure 9: Example Fragmented Packet (Packet 1)

In this payload the initial sequence number is 1000 and the timestamp is 12345. The Fragment type is set to 1, the number of packets field is set to 0, and as the payload is raw Vorbis data the VDT field is set to 0.

Packet 2:



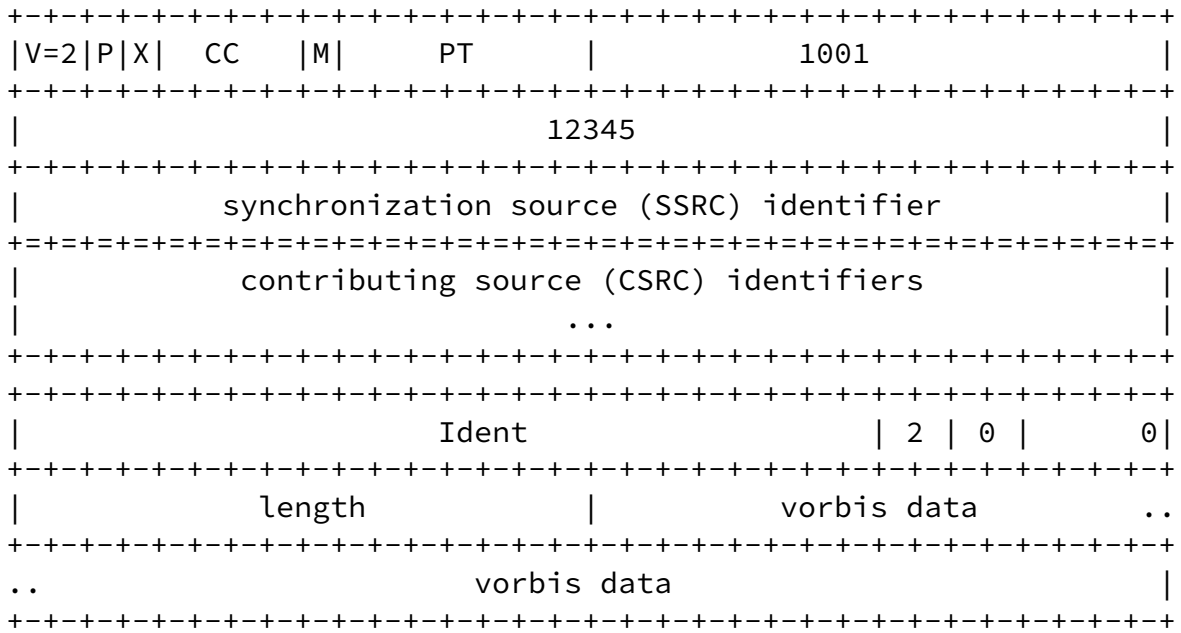


Figure 10: Example Fragmented Packet (Packet 2)

The Fragment type field is set to 2 and the number of packets field is set to 0. For large Vorbis fragments there can be several of this type of payloads. The maximum 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 payload.

Packet 3:

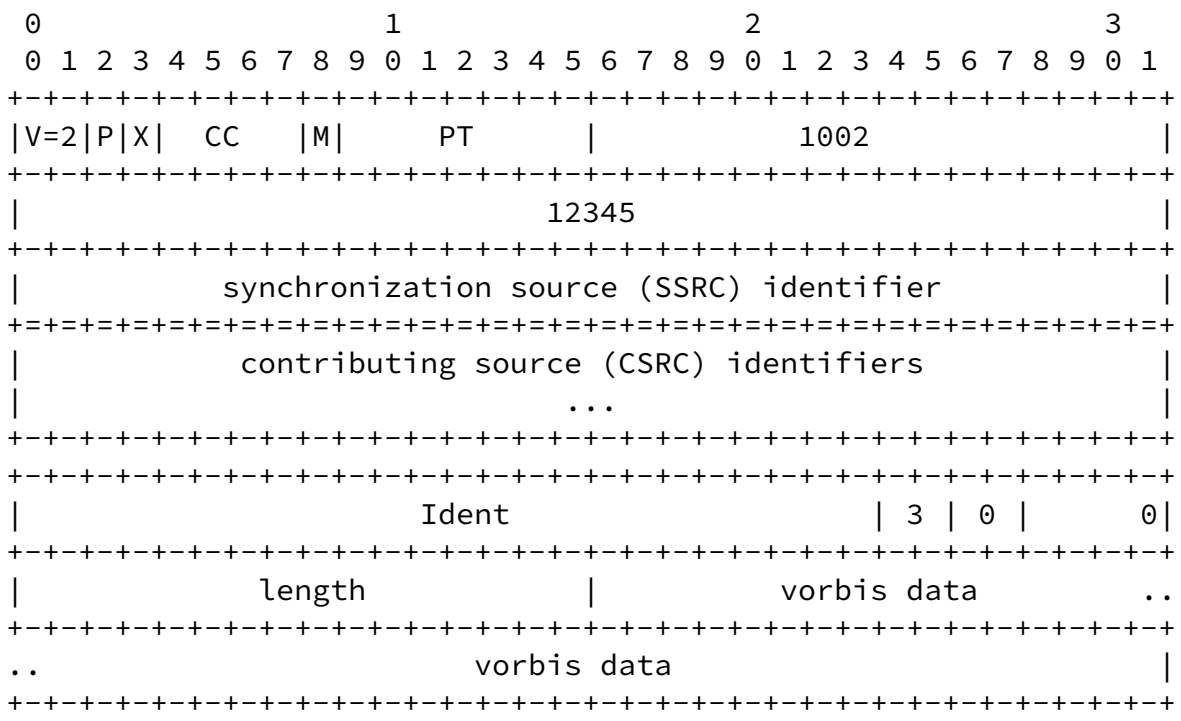


Figure 11: Example Fragmented Packet (Packet 3)

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

5.2. Packet Loss

As there is no error correction within the Vorbis stream, packet loss will result in a loss of signal. Packet loss is more of an issue for fragmented Vorbis packets as the client will have to cope with the handling of the Fragment Type. In case of loss of fragments the client MUST discard all the remaining Vorbis fragments and decode the incomplete packet. If we use the fragmented Vorbis packet example above and the first RTP payload is lost the client MUST detect that the next RTP payload has the packet count field set to 0 and the Fragment type 2 and MUST drop it. The next RTP payload, which is the final fragmented packet, MUST be dropped in the same manner. If the missing RTP payload is the last, the received two fragments will be kept and the incomplete Vorbis packet decoded.

Loss of any of the Configuration fragment will result in the loss of the full Configuration packet with the result detailed in the Loss of Configuration Headers (Section 3.3) section.

6. IANA Considerations

Type name: audio

Subtype name: vorbis

Required parameters:

rate: indicates the RTP timestamp clock rate as described in RTP Profile for Audio and Video Conferences with Minimal Control. [3]

channels: indicates the number of audio channels as described in RTP Profile for Audio and Video Conferences with Minimal Control. [3]

configuration: the base64 [9] representation of the Packed Headers (Section 3.2.1).

Encoding considerations:

This media type is framed and contains binary data.

Security considerations:

See Section 10 of RFC XXXX.

Interoperability considerations:

None

Published specification:

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

Ogg Vorbis I specification: Codec setup and packet decode. Available from the Xiph website, http://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 [2]

Author:

Luca Barbato

Change controller:

IETF AVT Working Group delegated from the IESG

6.1. Packed Headers IANA Considerations

The following IANA considerations refers to the split configuration Packed Headers (Section 3.2.1) used within RFC XXXX.

Type name: audio

Subtype name: vorbis-config

Required parameters:

None

Optional parameters:

None

Encoding considerations:

This media type contains binary data.

Security considerations:

See Section 10 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:

Vorbis encoded audio, 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 doesn't depend on the transport.

Author:

Luca Barbato

Change controller:

IETF AVT Working Group delegated from the IESG

7. SDP related considerations

The following paragraphs define the mapping of the parameters described in the IANA considerations section and their usage in the Offer/Answer Model [8]. In order to be forward compatible the implementation MUST ignore unknown parameters.

7.1. Mapping Media Type Parameters into SDP

The information carried in the Media Type specification has a specific mapping to fields in the Session Description Protocol (SDP) [5], which is commonly used to describe RTP sessions. When SDP is used to specify sessions the mapping are as follows:

- The type name ("audio") goes in SDP "m=" as the media name.
- The subtype name ("vorbis") goes in SDP "a=rtpmap" as the encoding name.
- The parameter "rate" also goes in "a=rtpmap" as clock rate.
- The parameter "channels" also goes in "a=rtpmap" as channel count.
- The mandated parameters "configuration" MUST be included in the SDP "a=fmtp" attribute.

If the stream comprises chained Vorbis files and all of them are known in advance, the Configuration Packet for each file SHOULD be passed to the client using the configuration attribute.

The port value is specified by the server application bound to the address specified in the c= line. The channel count value specified in the rtpmap attribute SHOULD match the current Vorbis stream or considered the maximum number of channels to be expected. The timestamp clock rate MUST be a multiple of the sample rate, different payload number MUST be used if the clock rate changes. The Configuration payload delivers the exact information, thus the SDP information SHOULD be considered as a hint. An example is found below.

7.1.1. SDP Example

The following example shows a basic SDP single stream. The first configuration packet is inlined in the SDP, other configurations could be fetched at any time from the URIs provided. The inline base64 [9] configuration string is folded in this example due to RFC line length limitations.

```
c=IN IP4 192.0.2.1
m=audio RTP/AVP 98
a=rtpmap:98 vorbis/44100/2
a=fmtp:98 configuration=AAAAAZ2f4g9NAh4aAXZvcmJpcwA...;
```

Note that the payload format (encoding) names are commonly shown in upper case. Media Type subtypes are commonly shown in lower case.

These names are case-insensitive in both places. Similarly, parameter names are case-insensitive both in Media Type types and in the default mapping to the SDP a=fmtp attribute. The a=fmtp line is a single line even if it is shown as multiple lines in this document for clarity.

7.2. Usage with the SDP Offer/Answer Model

The are no negotiable parameters. All the of them are declarative.

8. Congestion Control

The general congestion control considerations for transporting RTP data apply to vorbis audio over RTP as well. See the RTP specification [2] and any applicable RTP profile (e.g., [3]). Audio data can be encoded using a range of different bit rates, so it is possible to adapt network bandwidth by adjusting the encoder bit rate in real time or by having multiple copies of content encoded at different bit rates.

9. Example

The following example shows a common usage pattern that MAY be applied in such situation, the main scope of this section is to explain better usage of the transmission vectors.

9.1. Stream Radio

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 webjockey's voice, 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 listening to the content in a short time.

On join the client will receive the current Configuration necessary to decode the current stream 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 in-band method is unreliable, an out of band fallback is provided.

The client may choose to fetch the Configuration from the alternate source as soon as it discovers a Configuration packet got lost in-

band or use selective retransmission [13], 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.

10. Security Considerations

RTP packets using this payload format are subject to the security considerations discussed in the RTP specification [2], the base64 specification [9] and the URI Generic syntax specification [4]. Among other considerations, this implies that the confidentiality of the media stream is archieved 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.

11. Copying Conditions

The authors agree to grant third parties the irrevocable right to copy, use and distribute the work, with or without modification, in any medium, without royalty, provided that, unless separate permission is granted, redistributed modified works do not contain misleading author, version, name of work, or endorsement information.

12. Acknowledgments

This document is a continuation of draft-moffitt-vorbis-rtp-00.txt and draft-kerr-avt-vorbis-rtp-04.txt. The Media Type declaration is a continuation of draft-short-avt-rtp-vorbis-mime-00.txt.

Thanks to the AVT, Vorbis Communities / Xiph.Org Foundation including Steve Casner, Aaron Colwell, Ross Finlayson, Fluendo, Ramon Garcia, Pascal Hennequin, Ralph Giles, Tor-Einar Jarnbjo, Colin Law, John Lazzaro, Jack Moffitt, Christopher Montgomery, Colin Perkins, Barry Short, Mike Smith, Phil Kerr, Michael Sparks, Magnus Westerlund, David Barrett, Silvia Pfeiffer, Stefan Ehmann, Alessandro Salvatori. Politecnico di Torino (LS)^3/IMG Group in particular Federico Ridolfo, Francesco Varano, Giampaolo Mancini, Dario Gallucci, Juan Carlos De Martin.

13. References

13.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119.

[2] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for real-time applications", STD 64, RFC 3550.

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

[4] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986.

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

[6] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[7] McCann et al., J., "Path MTU Discovery for IP version 6", RFC 1981.

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

[9] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548.

[10] "Ogg Vorbis I specification: Codec setup and packet decode. Available from the Xiph website, http://xiph.org/vorbis/doc/Vorbis_I_spec.html".

13.2. Informative References

[11] Pfeiffer, S., "The Ogg Encapsulation Format Version 0", RFC 3533.

[12] "libvorbis: Available from the dedicated website, http://vorbis.com".

[13] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[14] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

Author's Address

Luca Barbato
Xiph.Org Foundation

EMail: lu_zero@gentoo.org
URI: http://xiph.org/

Full Copyright Statement

Intellectual Property

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.

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).