Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-avt-rtp-format-guidelines-00.ps

AVT Working Group
Mark Handley
ISI

December 16, 1997
Expires: June 1998

Guidelines for writers of RTP payload format specifications

Status of this Memo

Abstract

This document provides general guidelines aimed at assisting the authors of RTP Payload Format specifications in deciding on good formats. These guidelines attempt to capture some of the experience gained with RTP as it evolved during its development.

1 Introduction

This document provides general guidelines aimed at assisting the authors of RTP [1] Payload Format specifications in deciding on good formats. These guidelines attempt to capture some of the experience gained with RTP as it evolved during its development.

2 Background

RTP was designed around the concept of Application Level Framing (ALF), first described by Clark and Tennenhouse[2]. The key argument underlying ALF is that there are many different ways an application might be able to cope with misordered or lost packets. These range from ignoring the loss, to resending the missing data (either from a buffer or by regenerating it), and to sending new data which supersedes the missing data. The application only has this choice if transport protocol is dealing with data in "Application Data Units" (ADUs). An ADU contains data that can be processed out-of-order with respect to other ADUs. Thus the ADU is the minimum unit of error recovery.

The key property of a transport protocol for ADUs is that each ADU contains sufficient information to be processed by the receiver immediately. An example is a video stream, wherein the compressed video data in an ADU must be capable of being decompressed regardless of whether previous ADUs have been received. Additionally the ADU must contain "header" information detailing its position in the video image and the frame from which it came.

Although an ADU need not be a packet, there are many applications for which a packet is a natural ADU. Such ALF applications have the great advantage that all packets that are received can be processed by the application immediately.

RTP was designed around an ALF philosphy. In the context of a stream of RTP data, an RTP packet header provides sufficient information to be able to identify and decode the packet irrespective of whether it was received in order, or whether preceding packets have been lost. However, these arguments only hold good if the RTP payload formats are also designed using an ALF philosophy.

3 Guidelines

We identify the following requirements of RTP payload format specifications:

- A payload format should be devised so that in the presence of loss, the payload can still be decoded.
- Ideally all the contents of every packet should be possible to be decoded and played out irrespective of whether preceding packets have been lost or arrive late.

The first of these requirements is based on the nature of the internet. Although it may be possible to engineer parts of the internet to produce low loss rates through careful provisioning or the use of non-best-effort services, as a rule payload formats should not be designed for these special purpose environments. Payload formats should be designed to be used in the public internet with best effort service, and thus should expect to see moderate loss rates (for example, a 5% loss rate is not uncommon). Where payload formats do not make these assumptions, they should state this explicitly up front, and they will be considered special purpose payload formats, unsuitable for use on the public internet without special support from the network infrastructure.

The second of these requirements is more explicit about how RTP should cope with loss. If an RTP payload format is properly designed, every packet that is actually received should be useful. Typically this implies the following guidelines are adhered to:

- Packet boundaries should coincide with codec frame boundaries. Thus a packet should normally consist of one or more complete codec frames.
- A codec's minimum unit of data should never be packetised so that it crossed a packet boundary unless it is larger than the MTU.
- If a codecs frame size is larger than the MTU, the payload format must not rely on IP fragmentation. Instead it must define its own fragmentation mechanism. Such mechanisms may involve codec-specific information that allows decoding of fragments. Alternatively they might allow codec-independent packet-level forward error correction[3] to be applied that cannot be used with IP-level fragmentation.

In the abstract, a codec frame (i.e., the ADU or the minimum size unit that has semantic meaning when handed to the codec) can be of arbitrary size. For PCM audio, it is one byte. For GSM audio, a frame corresponds to 20ms of audio. For H.261 video, it is a Group of Blocks (GOB), or one twelfth of a CIF video frame.

For PCM, it does not matter how audio is packetised, as the ADU size is one byte. For GSM audio, arbitrary packetisation would split a 20ms frame over two packets, which would mean that is one packet were lost, partial frames in packets before and after the loss are meaningless. This means that not only were the bits in the missing packet lost, effectively additional bits in packets that used bottleneck bandwidth were also lost because the receiver must throw them away. Instead, we would packetise GSM by including several complete GSM frames in a packet; typically four GSM frames are included in current implementations. Thus every packet received can be decoded because even in the presence of loss, no incomplete frames are received.

The H.261 specification allows GOBs to be up to 3KBytes long, although most of the time they are smaller than this. It might be thought that we should insert a group of blocks into a packet when it fits, and arbitrarily split the GOB over two or more packets when a GOB is large. In the first version of the H.261 payload format, this is what was done. However, this still means that there are circumstances where H.261 packets arrive at the receiver and must be discarded because other packets were lost - a loss multiplier effect that we wish to avoid. In fact there are smaller units than GOBs in the H.261 bit-stream called macroblocks, but they are not identifiable without parsing from the start of the GOB. However, if we provide a little additional information at the start of each packet, we can re-instate information that would normally be found by parsing from the start of the GOB, and we can packetise H.261 by splitting the data stream on macroblock boundaries. This is a less obvious packetisation for H.261 than the GOB packetisation, but it does mean that a slightly smarter depacketiser at the receiver can reconstruct a valid H.261 bitstream from a stream of RTP packets that has experienced loss, and not have to discard any of the data that arrived.

An additional guideline concerns codecs that require the decoder state machine to keep step with the encoder state machine. Many audio codecs such as LPC or GSM are of this form. Typically they are loss tolerant, in that after a loss, the predictor coefficients decay, so that after a certain amount of time, the predictor error induced by the loss will disappear. Most codecs designed for telephony services are of this form because they were designed to cope with bit errors without the decoder remaining in permanent error. Just packetising these formats so that packets consist of integer multiples of codec frames may not be optimal, as although the packet received immediately after a packet loss can be decoded, the start of the audio stream produced will be incorrect (and hence distort the signal) because the decoder predictor is now out of step with the encoder. In principle, all of the decoder's internal state could be added using a header attached to the start of every packet, but for lower bit-rate encodings, this state is so substantial that the bit rate is no longer low bit rate. However, a compromise can usually be found, where a greatly reduced form of decoder state is sent in every packet, which does not recreate the encoders predictor precisely, but does reduce the magnitude and duration of the distortion produced when the previous packet is lost. Such compressed state is by definition, very dependent on the codec in question. Thus we recommend:

- Payload formats for encodings where the decoder contains internal data-driven state that attempts to track encoder state should normally consider including a small additional header that conveys the most critical elements of this state to reduce distortion after packet loss.

4 Summary

Designing packet formats for RTP is not a trivial task. Typically a detailed knowledge of the codec involved is required to be able to design a format that is resilient to loss, does not introduce loss magnification effects due to inappropriate packetisation, and does not introduce unnecessary distortion after a packet loss. We believe that considerable effort should be put into designing packet formats that are well tailored to the codec in question. Typically this requires a very small amount of processing at the sender and receiver, but the result can be greatly improved quality when operating in typical internet environments.

It is possible to design generic packetisation formats that do not pay attention to the issues described in this document, but such formats are only suitable for special purpose networks where packet loss can be avoided by careful engineering at the network layer, and are not suited to current best-effort networks.

References

[1] H.Schulzrinne, S.Casner, R.Frederick, V. Jacobson, ”Real-Time Transport Protocol”, RFC1899.

[2] D. Clark, D. Tennenhouse, ”Architectural Considerations for a New Generation of Network Protocols” Proc ACM Sigcomm 90.

[3] J. Nonnenmacher, E. Biersack, Don Towsley, “Parity-Based Loss Recovery for Reliable Multicast Transmission”, Proc ACM Sigcomm '97, Canne, France, 1997.