Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-avt-germ-00.ps
AVT WG
Mark Handley
ISI
November 11, 1998
Expires: May, 1999

GeRM: Generic RTP Multiplexing

Status of this Memo

Abstract

This document describes GeRM, an RTP payload format for generic multiplexing of multiple RTP streams.

This document is a product of the Audio/Video Transport (AVT) working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at rem-conf@es.net and/or the author.

1 Introduction

When RTP[1] is used for end-to-end communication, each RTP data stream in a session should be send separately to a different UDP port. This allows heterogeneous treatment of the streams by the network. For example, in a multimedia conference, we may be willing to pay to make an RSVP[2] reservation for the audio, but unable to reserve sufficient bandwidth for the video. Thus in the general case, we argue that multiplexing multiple RTP streams together should be avoided.

However, there are circumstances when this general rule may not make a great deal of sense. If a stream is very low bandwidth, but needs low latency, the overhead of RTP packetisation may be too large. On slow modem links this can be overcome by using IP/UDP/RTP header compression [3], but this is a hop-by-hop compression scheme, and so is unsuitable for congested high-speed backbone links.

MPEG 4 is an example of a codec that produces multiple elementary streams that comprise a single video stream. Many of these elementary streams are very low bandwidth. It makes little sense to packetise each of these elementary streams separately and send it to its own RTP/UDP port. Instead a network-aware multiplexing layer is required that can combine multiple elementary stream data units into a single RTP packet in a way that does not reduce resilience to packet loss[4].

Another example is that of IP telephony gateways. In such a gateway, incoming PSTN calls are packetised over RTP and transmitted to a remote gateway, where they are turned back into PSTN calls again. Between any pair of gateways there may be many simultaneous telephone calls. If a relatively low bitrate codec is used such as GSM (approx 14Kbps), each of these flows then gains its own IP, UDP and RTP headers comprising 40 bytes. With 20ms packetisation, the overhead is over 100%. In this case, many (if not all) of the flows expect the same network service. The header overhead can be significantly reduced if multiple unrelated flows are multiplexed together into a single RTP packet.

In both these cases and other similar ones, we can design specific multiplexing protocols that satisfy one particular problem domain. Rather than do this, we propose a multiplexing protocol that attempts to be generic. Any pair of RTP flows with the same source and destination may be multiplexed together. The degree of compression depends on the similarity of the two flows, but the per-flow overhead is always less than a single RTP header (without IP or UDP), and is typically much better.

2 Specification

The approach taken in GeRM is similar to that taken with IP/UDP/RTP header compression, in that only differences between one packet and the next are encoded. However, unlike IP/UDP/RTP header compression, GeRM does this by only encoding the differences between the RTP headers of the different payloads in the same multiplexed packet, and all RTP header state is reinitialised in each new packet. As a result GeRM can function effectively across multiple network hops.

The basic model is that a single IP packet contains multiple RTP headers each followed by its own payload. Each of these RTP headers followed by its payload is referred to as a *sub-packet*. GeRM now compresses each sub-packet header, so that fields which are predictable between one sub-header and the next sub-header within the same packet are not sent.

Each multiplexed RTP packet has a full RTP header which contains the SSRC, Sequence number, Timestamp, etc corresponding to the first sub-packet payload, but the RTP payload type field is set to a value indicating this is a GeRM packet. The first sub-packet header will compress out completely except for the payload-type field and length because the full RTP header and the sub-packet header only differ in the payload-type.

The second sub-packet header is then encoded based on predictable differences between the original RTP header for that sub-packet and the original RTP header for the first sub-packet.

The third sub-packet header is then encoded off of the original RTP header for the second sub-packet and so forth.

A regular RTP header has the following format:

```
 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      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A GeRM header consists of one byte followed by any RTP fields that are not predictable from the previous header. The parts of the RTP header corresponding to the bits of the GeRM header are as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V |P|X|  CC   |M|     PT      |       sequence number         |
|0 0|0|0|0 0 0 0|1|2 2 2 2 2 2 2|3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
|4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
|6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6|
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                        not compressed                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The GeRM Header is one byte:

```
 0  1  2  3  4  5  6  7
+--+--+--+--+--+--+--+--+
|B0|B1|B2|B3|B4|B5|B6|B7|
+--+--+--+--+--+--+--+--+
```

The meaning of these bits is:

B0:

- zero indicates that the first byte of the original RTP header remains unchanged from the original RTP header in the previous subpacket (or outer RTP header if there's no previous sub-packet in this packet). I.e, V, CC and P are unchanged.
- one indicates that the first byte (byte 1) of the original RTP header immediately follows the GeRM header.

B1: Contains the marker bit from the sub-packet's RTP header.

B2:

- zero indicates that the payload type remains unchanged.
- one indicates that the payload type field follows the GeRM header and any byte 0 header that may be present. Although PT is a seven bit field, it is added as an eight bit field. Bit 0 of this byte MUST be zero.

B3:

- zero indicates that the sequence number remains unchanged.
- one indicates that the 16 bit sequence number field follows the GeRM header and any byte 1 or PT header that may be present.

B4:

- zero indicates that the timestamp remains unchanged.
- one indicates that the 32 bit timestamp field follows the GeRM header and any byte 1, PT or sequence number header that may be present.

B5:

- zero indicates that the most significant 24 bits of the SSRC remain unchanged.
- one indicates that the most significant 24 bits of the SSRC follows the GeRM header and any byte 1, PT, sequence number or timestamp field that may be present.

B6:

- zero indicates that the least significant 8 bits of the SSRC are one higher than the preceding SSRC.
- one indicates that the least significant 8 bits of the SSRC follows the GeRM header and any byte 1, PT, sequence number, timestamp or MSB SSRC header fields that may be present.

B7:

- zero indicates that the sub-packet length in bytes (ignoring the sub-packet header) is unchanged from the previous sub-packet.
- one indicates that the sub-packet length (ignoring the sub-packet header) follows all the other GeRM headers as an 8-bit unsigned integer length field.

Any CSRC fields present in the original RTP header then follow the GeRM headers.

3 Examples

In this section we attempt to characterise the likely behaviour of GeRM in some typical circumstances.

3.1 Arbitrary Streams, Same Payload Type

Five RTP streams that originate at separate RTP sources (with SSRCs SSRC1 to SSRC5, sequence numbers SEQ1 to SEQ5, and timestamps T1 to T5) are being multiplexed together. They each use GSM compression, and the GSM codec uses fixed size frames.

The compound packet is as follows:

```
IP Header
UDP Header
RTP Header, V=0, P=0, CC=0, PT=>GERM, SEQ=SEQ1, TS=T1, SSRC=SSRC1
GeRM Header, B0=0, B1=M1, B2=1, B3=0, B4=0, B5=0, B6=0, B7=1
  8-bit PT=>GSM
  8-bit length = length of GSM frame
GSM payload 1
GeRM Header, B0=0, B1=M2, B2=0, B3=1, B4=1, B5=1, B6=1, B7=0
  16-bit Sequence Number SEQ2
```

```
    32-bit Timestamp TS2
    24+8 bit SSRC SSRC2
  GSM payload 2
  ...
  GeRM Header, B0=0, B1=M5, B2=0, B3=1, B4=1, B5=1, B6=1, B7=0
    16-bit Sequence Number SEQ5
    32-bit Timestamp TS5
    24+8 bit SSRC SSRC5
  GSM payload 5
```

The overhead is 40 bytes (IP+UDP+RTP) + 3 bytes (first sub-header) + 11 bytes (each subsequent sub-header), or a total of 87 bytes, as opposed to 200 bytes for separate RTP packets.

In some cases, having a multiplex stream sequence number in the outer RTP packet (rather than the first payload sequence number) might be desirable. This might be the case if we wish to add packet-level FEC to the multiplexed stream. In such a case, the sequence number of sub-packet 1 does not compress out, adding a further two bytes overhead.

3.2 Cooperating PSTN-IP gateways

If several RTP streams coded with the same codec are ordinating at a PSTN->IP gateway and all terminate at the same IP->PSTN gateway, and if we assume that an out-of-band signalling mechanism is used to communicate SSRC information at call setup time, then we can achieve significantly better compression.

To do this we algorithmically generate the SSRC rather than allocating it randomly as specified in the RTP specification. This is acceptable in this context because only the remote gateway will ever see the SSRC.

As consecutive flows arrive, they are given consecutive SSRCs, which in any event must be communicated as part of the call setup mechanism. All the flows are digitised and compressed at the same time, so they share a common clock and hence common timestamps. If no silence suppression is performed, the sequence numbers can be consecutive too, but we do not assume this.

As flows terminate, they will leave gaps in the SSRC space. New flows are then allocated the now unused SSRCs to attempt to keep the SSRC space as contiguous as possible.

For the sake of example, we assume we have SSRCs 1 to 3 and 5 to 10 in use, and that the flows with SSRC 5, 7 and 8 are being silence suppressed. This leaves us with flows 1,2,3,6,9 and 10 to transmit.

```
  IP Header
  UDP Header
  RTP Header, V=0, P=0, CC=0, PT=>GERM, SEQ=SEQ1, TS=T1, SSRC=SSRC1
  GeRM Header, B0=0, B1=M1, B2=1, B3=0, B4=0, B5=0, B6=0, B7=1
    8-bit PT=>GSM
    8-bit length = length of GSM frame
  GSM payload 1
  GeRM Header, B0=0, B1=M2, B2=0, B3=1, B4=0, B5=0, B6=0, B7=0
    16-bit Sequence Number SEQ2
  GSM payload 2
  GeRM Header, B0=0, B1=M3, B2=0, B3=1, B4=0, B5=0, B6=0, B7=0
```

```
    16-bit Sequence Number SEQ3
  GSM payload 3
  GeRM Header, B0=0, B1=M6, B2=0, B3=1, B4=0, B5=0, B6=1, B7=0
    16-bit Sequence Number SEQ6
    8-bit LSByte of SSRC 6
  GSM payload 6
  GeRM Header, B0=0, B1=M9, B2=0, B3=1, B4=0, B5=0, B6=1, B7=0
    16-bit Sequence Number SEQ9
    8-bit LSByte of SSRC 9
  GSM payload 9
  GeRM Header, B0=0, B1=M10, B2=0, B3=1, B4=0, B5=0, B6=0, B7=0
    16-bit Sequence Number SEQ10
  GSM payload 10
```

Thus the overhead is 40 bytes for the IP/UDP/RTP, 3 bytes for sub-header 1, 4 bytes each for sub-headers 2, 3, and 10, and 5 bytes for sub-headers 6 and 9. This totals 65 bytes against 240 bytes for separate IP/UDP/RTP headers per flow. Typically each new flow being included in the packet will require 4 to 5 bytes of overhead in addition to the compressed data itself.

We might envisage ways in which sequence numbers of flows can also be manipulated as a flow returns from silence suppression (step the sequence number to match that of the flow with preceding SSRC) if we are sure that the flow will be removed from RTP at the next destination. This would reduce the per-flow overhead to between 1 and 5 bytes depending on the effectiveness of this mapping. Whether this is worth pursuing is an open issue that providers may consider. It MUST NOT be done unless the destination knows to expect such behaviour and not treat it as loss.

Appendix A: Author's Address

Mark Handley
Information Sciences Institute,
University of Southern California,
c/o MIT Laboratory for Computer Science,
545 Technology Square,
Cambridge, MA 02139,
United States
electronic mail: mjh@isi.edu

References

[1] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson "RTP: A Transport Protocol for Real-Time Applications" RFC 1889.

[2] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205

[3] S. Casner, V. Jacobson, “Compressing IP/UDP/RTP Headers for Low-Speed Serial Links”, Internet Draft.

[4] M. Handley, “Guidelines for writers of RTP payload format specifications”, Internet Draft.