Common Generic RTP Payload Format

Anders Klemets

Microsoft

RTP Payload Formats

- ✓ Each codec bitstream requires an RTP P.F.
  - Defines how the bitstream is encapsulated in RTP.
  - May define a Payload Format Header that should be included in the RTP packet.
  - May use knowledge of the codec to provide resiliency against lost RTP packets.
    - Redundant information.
    - Independently decodable packets.

Problems

- ✓ Servers need a P.F. implementation for each supported codec bitstream.
  - Each new codec requires upgrading the server.
  - Server administrator must trust content provider and software vendor.
- ✓ Extensible file formats that use many codecs.
  - Examples: ASF, QuickTime, MPEG-4 File Format
- ✓ Lengthy standardization process.
  - What if I want to use my new codec *now*?

Generic RTP Payload Formats

- ✓ A codec independent RTP P.F. is needed
- ✓ Multiple file format dependent proposals:
  - QuickTime
  - ASF
  - MPEG-4 File Format
- ✓ The Common Generic RTP Payload Format:
  - codec independent
  - file format independent
  - provides common framework for specialized P.F.'s

Overview of features

- ✓ Support for fragmentation
  - codec-aware fragmentation
  - codec-unaware fragmentation
- ✓ Support for grouping
  - Fragments may be grouped, allowing fixed size packets
- ✓ Extensibility
  - Uses grouping mechanism
  - Extra timestamps, flags, duration fields, etc., go here

Codec-unaware fragmentation

- ✓ Simple "network layer" fragmentation.
- ✓ If a fragment is lost, all other fragments of the same "PDU" must be discarded.
- ✓

S=1, M=0 : First fragment
S=1, M=1 : Last fragment
S=0, M=0 : Other fragment

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|S|     0     |                   PDU                         .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Codec-aware fragmentation

✓ Fragment boundaries chosen by "app. layer"
  - May allow a partial "App. Unit" to be decoded

✓ OFFSET field gives placement of fragment

✓ FRAG field gives fragment sequence number
  - Separates fragments of different "App. Units."

S: Shift OFFSET 8 bits to the left
FRAG: Fragment number (<> 0)
OFFSET: Byte offset of fragment into App. Unit

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|S|   FRAG    |                   OFFSET                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                              PDU                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


Nested fragmentation

✓ Codec-aware and codec-unaware fragmentation can be combined.

✓ Example:
  1. Codec-aware fragments are read from a file.
  2. Some of the fragments have a size that is > MTU.
  3. Codec-unaware fragmentation is applied on the fragments that exceed the MTU size.

Grouping (bundling)

✓ RTP packets may contain multiple PDU's.
  - TIMESTAMP DELTA field allows for different presentation times.

✓ Fragments (of both kinds) may be grouped.
  - Allows for constant size RTP packets.

✓ Grouped elements can be tagged as "Extension Data"
  - Allows arbitrary extension headers for each PDU.

Extension Data

✓ Allows arbitrary extensions (specify thru SDP?)

✓ Example of extensions that can be ignored:
  - Send Time timestamp
  - Duration field
  - Key Frame flag, etc.

✓ Extensions that cannot safely be ignored:
  - Multiplexing of multiple logical streams into one RTP packet.
    - C.f. FlexMux in MPEG-4

Grouping example 1

G=1: Indicates presence of Grouping fields
E=1: Indicates Extension Data

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|S|   FRAG    |                   OFFSET                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|0|          LENGTH           |                                .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                                .
.                     PDU (Last Fragment)                        .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|0|          LENGTH           |                                .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                                .
.                      Extension Data                            .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|B|          LENGTH           |        TIMESTAMP DELTA         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                     PDU (First Fragment)                       .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

LENGTH: Indicates fragmentation
B: Indicates fragment type

Grouping example 2

G=1: Indicates presence of Grouping header
E=1: Indicates Extension Data
B=1: Indicates that no Grouping header follows

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|S|      0    |1|1|            LENGTH           |              .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+              .
.                      Extension Data                            .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                              PDU                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Overhead per RTP packet

Bytes	Features
1	1 PDU, codec-unaware fragmentation
3	1 PDU, codec-unaware fragmentation, extension data
4	1 PDU, codec-aware fragmentation
7	2 PDU's with different TS, codec-unaware fragmentation
11	2 PDU's with different TS, codec-unaware fragmentation, 2 extension data fields

Conclusion

A generic RTP payload format with:

✓ minimalist set of features
- features interact, can be combined
- attempt to cover most usage scenarios

✓ low-overhead packet format

✓ ease of extensibility
- the *Common* format can be extended to a *Specialized* format