Internet Engineering Task Force
Internet Draft
ietf-avt-rtp-mpeg4-00.txt
March 09, 1998
Expires: September 09, 1998

Audio/Video Transport Working Group
Herpel-Thomson/Balabanian-Nortel
Basso-AT&T/Civanlar-AT&T/Hoffman-Sun
Speer-Sun/Schulzrinne-Columbia U.

RTP payload format for MPEG-4 Elementary Streams

ABSTRACT

This memorandum describes a scheme to encapsulate MPEG-4 Elementary Streams into RTP packets. Two approaches are described. The first allows to map one MPEG-4 Elementary Stream to one RTP session, for maximum compatibility to the design principles of RTP.
The second responds to the observation that MPEG-4 applications may well consist of such a large number of streams that the encapsulation of a bundle of Elementary Streams in one RTP session is required.

This specification is a product of the Audio/Video Transport working group within 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 authors.

1 Introduction

MPEG-4 is a recent standard from ISO/IEC for the coding of natural and synthetic audio-visual data in the form of audiovisual objects that are arranged into an audiovisual scene by means of a scene description [1][2][3][4]. This memorandum specifies how MPEG-4 encoded data streams are mapped into the real-time transport protocol (RTP)[5].

This would allow the use of RTP tools to monitor MPEG-4 delivery performance, the use of RTP mixers to combine MPEG-4 streams received from multiple end-systems into a set of consolidated streams for multicasting and the use of RTP translators.

1.1 Overview of MPEG-4 End-System Architecture

Figure 1 below shows the general architecture of MPEG-4 terminals. The Compression Layer processes individual audio-visual media streams without regard to delivery technologies. The compression schemes in MPEG-4 achieve efficient encoding over a wide range from Kbps to multiple Mbps. The MPEG-4 compression schemes are defined in the ISO/IEC specifications 14496-2 and -3 [2][3]. The media content at this layer is organized in Elementary Streams.

The MPEG-4 Systems specification, ISO/IEC 14496-1 [1], defines the concepts needed to describe the relations between Elementary Streams in a way that allows to create distributed, yet integrated, content presentations and to synchronize the streams. This part of the specification is both media unaware and delivery technology unaware.

The hierarchical relations, location and properties of Elementary Streams of a presentation are described by a dynamic set of Object Descriptors (ODs). Each Object Descriptor groups one or more Elementary Stream Descriptors that all refer to a single content item (media object). Hence, multiple alternative or hierarchical representations of each content item can be indicated.

Object Descriptors are themselves conveyed through one or more Elementary Streams. A complete set of ODs can be seen as an MPEG-4 resource or session description at a stream level. The resource description may itself be hierarchical, accomplished by creating more than one Elementary Stream conveying Object Descriptors.

The session description is accompanied by a dynamic scene description (Binary Format for Scene, BIFS), again conveyed through one or more Elementary Streams. At this level, content is identified in terms of media objects. The spatiotemporal location of each object is defined by BIFS. The media content of the object, if synthetic and static, will also be described by BIFS, while natural and animated synthetic objects may refer to an Object Descriptor that points to one or more Elementary Streams that carry the coded representation of the object or its animation data.

By conveying the session (or resource) description as well as the scene (or content composition) description through their own Elementary Streams it is made possible to change portions of the content composition and the number and properties of media streams that carry the media content separately and dynamically at well known instants in time.

A homogeneous encapsulation of Elementary Streams carrying media or control (ODs, BIFS) data is defined by the Access Unit (AU) Layer that primarily serves to convey the synchronization between streams. The AU Layer organizes the Elementary Streams in Access Units, the smallest elements that can be attributed individual timestamps. Integer or fractional AUs are then encapsulated in AU Layer PDUs (AL-PDU). All consecutive data of one stream at this layer is called an AL-packetized Elementary Stream. The interface between the compression layer and the AU Layer is called Elementary Stream Interface (ESI). The ESI is informative.

The Delivery Layer in MPEG-4 consists of the Delivery Multimedia Integration Framework defined in ISO/IEC 14496-6 [4]. This layer is media unaware but delivery technology aware. It provides transparent access to and delivery of content irrespective of the technologies used. The interface between the AU Layer and DMIF is called DMIF Application Interface (DAI). It offers content location independent procedures for establishing MPEG-4 sessions and access to transport channels.

```
  media aware         +-----------------------------------------+
  delivery unaware    |           COMPRESSION LAYER             |
  14496-2 Visual      |streams from as low as Kbps to multi-Mbps|
  14496-3 Audio       +-----------------------------------------+  Elementary
                                                                   Stream
  ===============================================================Interface
                                                                   (ESI)
                     +-------------------------------------------+
  media and          |               SYSTEMS LAYER               |
  delivery unaware   | manages elementary streams, their synch-  |
  14496-1 Systems    | ronization and hierarchical relations     |
                     +-------------------------------------------+ DMIF
                                                                   Application
  ===============================================================Interface
                                                                   (DAI)
                     +-------------------------------------------+
  delivery aware     |              DELIVERY LAYER               |
  media  unaware     |provides transparent access to and delivery|
  14496-6 DMIF       | of content irrespective of delivery       |
                     |                technology                 |
                     +-------------------------------------------+
```

Figure 1: General MPEG-4 terminal architecture

1.2 MPEG-4 Elementary Stream Data Packetization

Figure 2 below shows the MPEG-4 System data plane. For ease of explanation the encoding side is described next. The Elementary Streams from encoders are fed into the Access Unit Layer with indications of AU boundaries, Random Access Points, Desired Composition Time and the current time.

The Access Unit Layer fragments the elementary streams into AL-PDUs each containing a header which encodes information conveyed through the ESI. If the AU is larger than a AL-PDU fragment then a subsequent PDU is generated with a smaller header until the complete AU is packetized into AL-PDU fragments.

The syntax of the Access Unit Layer is not fixed but can be adapted to the needs of the stream to be transported. This includes the possibility to select presence or absence of individual syntax elements as well as configuration of their length in bits. The configuration for each individual stream is conveyed in an ALConfigDescriptor which is an integral part of the Elementary Stream Descriptor for this stream.

The AL-packetized streams having the same priority and transport QoS requirements (as indicated through the DAI interface signaling) may then optionally be multiplexed in FlexMux Streams in order to fill out the bandwidth on the network and reduce latency for each stream. Finally, as figure 2 shows, the AL-packetized stream or the FlexMux stream is passed on to any transport layer, generically called TransMux.

```
                 E N C O D E R S  and D E C O D E R S
                          (elementary streams)
      ^    ^    ^        ^    ^        ^         ^
    ~~|~~~~|~~~~|~~~~~~~~|~~~~|~~~~~~~~|~~~~~~~~~|~~~~~~~~ESI~~~
      v    v    v        v    v        v         v               \
    +---++---++---+    +---++---+    +---+     +---+ Access Unit  |ISO/IEC
    |AL ||AL ||AL |    |AL ||AL |    |AL |     |AL |   Layer      |14496-1
    +---++---++---+    +---++---+    +---+     +---+              |MPEG-4
      ^    ^    ^        ^    ^        ^         ^                |Systems
      |    |    |        |    |        |         |               /
   ===|====|====|========|====|========|=========|==================DAI=
      v    v    v        v    v        v         |               \
   +---------------+   +---------+ +-------+     |\_______________|_ AL-packetized
   |   FlexMux     |   | FlexMux | |FlexMux|     |                |  Stream
   +---------------+   +---------+ +-------+     |  FlexMux Layer |
           ^                ^          ^         |                |
           | __ FlexMux-    |          |         |                |
           |/    Stream     |          |         |                |
           |                |          |         |                |
   ........v................v..........v.........v..............  |ISO/IEC
                                                                  |14496-6
     +---+   +-----+   +----+   +----+  +-----+   +---+           |DMIF
     |RTP|   |MPEG2|   |AAL5|   |AAL2|  |H.223|   |DAB|   TransMux|
     |UDP|   | TS  |   |ATM |   |ATM |  |PSTN |   |mux|etc., Layer|
     |IP |   |     |   |    |   |    |  |     |   |   |           |
     +---+   +-----+   +----+   +----+  +-----+   +---+           |
       ^        ^         ^        ^       ^        ^            /
    ~~~|~~~~~~~~|~~~~~~~~~|~~~~~~~~|~~~~~~~|~~~~~~~~|~~~~~~DNI~~~
       v        v         v        v       v        v
                        (TransMux streams)
                         N E T W O R K S
```

Figure 2: The MPEG-4 System Data Plane

MPEG does not restrict the permissible transport protocol stacks. Just the properties that such a protocol stack (the ‘TransMux Layer’) should have are specified by MPEG.

2 Encapsulation of packetized MPEG-4 data in RTP packets

2.1 Analysis of requirements

The RTP specification recommends using a separate session for each media stream. So, a straightforward use of RTP for MPEG-4 payloads may require one RTP session for each Elementary Stream, i.e., one or more streams per audiovisual object. Since a typical MPEG-4 session may involve a large number of objects, that may be as high as a few hundred, this approach is not always practical. Allocating and controlling hundreds of destination addresses for each MPEG-4 session will pose insurmountable session administration problems. The input/output processing overhead at the end-points will be extremely high also. On the other hand, creating a single payload for the entire object collection defies the highly valued object based scalability property of MPEG-4 for the Internet applications.

Therefore, one RTP payload type for MPEG-4 data should allow selective bundling of several objects or Elementary Streams, respectively. The bundling technique needs to address the following requirements:

a) The determination and announcement of the object bundles should be dynamic and may be done by the sender or the receiver(s) of an MPEG-4 session through non-RTP means. The required descriptors will be defined by DMIF and may either be encapsulated in a DMIF-defined protocol or may be part of an SDP [7] session description. They might be conveyed through native signaling, using e.g. SIP [8] INVITE or RTSP [9] DESCRIBE methods.

b) Once an object bundle is defined, it is considered to be a single payload type 'MPEG-4 FlexMux Stream' for RTP purposes and thus identified as a single RTP session.

Given that there are also applications with fewer objects and, hence, Elementary Streams, a second RTP payload type for MPEG-4 data should allow efficient encapsulation of a single Elementary Stream:

c) There need to be a payload type for 'MPEG-4 Single Stream'.

d) In general, an AL-PDU does not contain information about its size. Therefore an RTP packet in 'MPEG-4 Single Stream' mode should contain exactly one AL-PDU since, in that case, its length can be derived from the packet length indication of the underlying protocol, usually UDP/IP.

e) Different from other RTP payload formats both payload formats do not expose whether the payload is video or audio. This is no disadvantage since it is assumed that subsets of MPEG-4 streams of a presentation are identified by means of descriptive information, like stream priority and object dependencies, conveyed in Object Descriptors and DMIF descriptors, and not by the payload type.

2.2 MPEG-4 Single Stream Payload Type

For this payload type a single MPEG-4 AL-PDU is mapped into one RTP packet. For an optimized encapsulation, the redundancies between the AL-PDU header and the RTP header are removed as far as possible.

The AL syntax is flexible and may vary from stream to stream, therefore the RTP packer and unpacker processes need to have access to the ALConfigDescriptor for this stream that determines the variable syntax elements.

Furthermore it is assumed that the MPEG-4 Access Unit Layer packer is aware of the path-MTU (Maximum Transfer Unit) of the network. Hence, the AL is responsible for splitting Access Units in appropriate AL-PDUs that do not lead to transport packets that are larger than the path MTU size. This includes that such an AL entity is responsible for re-packetizing Access Units when an AL-packetized stream is read from a file that has assumed a different path MTU size upon creation.

An MPEG-4 AL-packetized Elementary Stream that does not respect the path-MTU of the underlying transport network and that cannot be re-packetized in an AL-aware manner for any reason has to use the FlexMux Stream payload type that does define means for fragmentation on the RTP layer.

Note: An AU corresponds to 'Application Data Unit' in ALF. An AL-PDU which carries less than a complete AU may be a 'stupid' fragment made for the sole purpose to adapt packet size to some network requirements. Therefore AL-PDUs never have to be fragmented, since the AL packer (that is closely linked to the RTP packer for the purpose of our work here) is path MTU aware. The fallback solution is expressed in the previous paragraph.

The Single Stream Payload type eliminates redundant transmission of sequence numbers and timestamps between RTP header and AL-PDU header. In conjunction with RTP header compression [6] it therefore allows relatively low overhead packetization.

2.2.1 RTP Header usage

The payload type (PT) shall be set to the value identified for ‘MPEG-4 Single Stream’.

The marker (M) bit shall be set to one for each AL-PDU that is the last AL-PDU of a fragmented Access Unit. Such an AL-PDU is identified either by
* accessUnitEndFlag=1, if this flag is present in the header, or
* looking ahead that the subsequent AL-PDU has accessUnitStartFlag=1 or
* knowing from ALConfigDescriptor that each AL-PDU contains a complete Access Unit.

Note: M is used as indicator for fragmentation that occurs on the AU Layer. This may be seen as not being strictly the same as the original RTP intention to use it to signal the end of a presentation unit (e.g. a video frame). However, for the purpose of this document an AU should be seen as presentation unit.

The extension (X) bit shall be set to zero.

The sequence number shall be derived from the value encountered in the sequenceNumber field of the AL-PDU, if present, incremented by a constant random offset. If sequenceNumber has less than 16 bit length, the MSBs shall initially be filled with a random value that is incremented by one each time the sequenceNumber value of the AL-PDU returns to zero. As an exception, the MSBs shall not be incremented if the value sequenceNumber=0 is encountered in multiple consecutive AL-PDUs since this indicates a deliberate duplication of the AL-PDU. The sequenceNumber shall then be removed from the AL-PDU header by bit-shifting the subsequent header elements towards the beginning of the AL-PDU header. When unpacking the RTP packet this process can be reversed with the knowledge of the ALConfigDescriptor.
If no sequenceNumber field is configured for this stream, acc. to the ALConfigDescriptor, then the RTP packer shall generate its own sequence numbers, starting at a random value.

The timestamp shall be set to the value encountered in the compositionTimeStamp field of the AL-PDU, if present. If compositionTimeStamp has less than 32 bits length, the MSBs of timestamp shall be set to zero.
If compositionTimeStamp has more than 32 bits length, ‘MPEG-4 Single Stream’ payload type can not be used and ‘MPEG-4 FlexMux Stream’ must be used instead for this stream.
In case compositionTimeStamp is not present in this AL-PDU, but has been present in a previous AL-PDU, this same value shall be taken again as timestamp.
The compositionTimeStamp, if present, shall then be removed from the AL-PDU header by bit-shifting the subsequent header elements towards the beginning of the AL-PDU header. When unpacking the RTP packet this process can be reversed with the knowledge of the ALConfigDescriptor and by evaluating the compositionTimeStampFlag.
The AL-PDU header also allows a decodingTimeStamp to be present, if the decoding time is different from the composition time. For those AL-PDUs in which only a decodingTimeStamp or both, composition and decodingTimeStamp are present in the AL-PDU header the above procedure shall be applied to the decodingTimeStamp, leaving the compositionTimeStamp unaffected.
If compositionTimeStamp is never present in AL-PDUs for this stream, acc. to the ALConfigDescriptor, the RTP packer shall convey a reading of a local clock at the time the RTP packet is sent.

Note: Timestamps should start at a random value for security reasons. Since they are copied from the AL, this would involve changing the remaining timestamps in the AL-PDU Header as well. This includes changing OCR timestamps that may have a different accuracy. Therefore it seems to be difficult to satisfy the random start value constraint.

An SSRC value for each RTP session shall be selected randomly, as described in RFC1889 [5].

2.2.2 RTP Payload

The RTP Payload consists of one single AL-PDU, including the AL-PDU header, omitting the sequenceNumber and decodingTimeStamp fields, as specified in the previous section. If the resulting, smaller, AL-PDU header consumes a non-integer number of bytes, zero padding bits shall be inserted to byte-align the AL-PDU payload.

Note: 32 bit alignment of AL-PDU payload would be desirable for processing efficiency with high bitrate streams, while for low bitrate streams unfortunately it just increases the overhead.

```
 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         |  RTP
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |  Header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:            contributing source (CSRC) identifiers             :
:                             ....                              :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|AL-PDU Header (variable # of bytes)             |              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |  RTP
|                                                               |
|       AL-PDU Payload (byte aligned)                           |  Payload
|                                                               |
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               :...........RTP padding (opt.)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The operation of the RTP unpacker for this payload type can be visualized by the figure below. Note that an implementation may as well integrate RTP parsing and AL-PDU regeneration in one unit.

```
    Natural/Synthetic Audio & Visual data,
       Scene Description & OD streams

     ^     ^     ^                     ^
     |     |     |                     |
     |     |     |                     |
   +---+ +---+ +---+                 +---+
   |AL | |AL | |AL |.  .  .  .  .  . |AL |
   +---+ +---+ +---+                 +---+
     ^    ^      ^                     ^
     |    |      |                     |
   ==|====|======|=====================|===DAI===
     |    |      |                     |
   +---------------------------------------+
   |        Regenerate AL-PDUs             |
   +---------------------------------------+
    MPEG-4  ^   Time  ^ Sequence^
   Payloads |  Stamps |  Numbers|
            |         |         |
          +------------------------+
          |      RTP Parser        |
          +------------------------+
                      ^
                      I
                      I
                      I
```

2.3 MPEG-4 FlexMux Stream Payload Type

This payload type is intended for use if the number of Elementary Streams in an MPEG-4 application are too high for the Single Stream payload type to be efficient, i.e., the number of RTP packets per second as well as the associated processing for the high number of RTP sessions becomes unmanageable. In this case multiple MPEG-4 Elementary Streams have to be mapped into one RTP session.

Therefore this payload type defines how data from multiple streams can be aggregated in one RTP packet, encapsulated in MPEG-4 FlexMux-PDUs as specified in [1]. Notably this enables the use of FlexMux MuxCode mode that allows to merge AL-PDUs from different streams in a pre-agreed sequence and, hence, does not need any further overhead to label these AL-PDUs.

This payload type is intended to carry an integer number of FlexMux-PDUs per RTP packet, however, it also defines a means to support fragmentation of large FlexMux-PDUs in case the MPEG-4 Access Unit Layer packer is not aware of the path-MTU of the network.

2.3.1 RTP Header usage

The payload type (PT) shall be set to the value identified for 'MPEG-4 FlexMux Stream'.

The marker (M) bit shall be zero if the payload starts with the beginning of a FlexMux-PDU. It shall be one if the payload contains the continuation of a FlexMux-PDU. The FlexMux-PDU length indication allows to know whether such an RTP packet contains the last fragment of a fragmented FlexMux-PDU.

The extension (X) bit shall be set to zero.

The RTP packer shall generate sequence numbers, incrementing monotonically by one, for each RTP packet generated in order. For the initial RTP packet of a session a random value shall be chosen for sequence number.
If two packets with the same sequence number are received immediately after each other, a duplication of the packet shall be assumed and the later packet may be discarded.
If the sequence numbers are discontinuous, packet loss shall be assumed as soon as the timestamp of the packet after the discontinuity requires delivery of that packet. The MPEG-4 Elementary Streams encapsulated in the FlexMux Stream that forms the RTP payload need to implement their own AL-PDU loss detection using the sequenceNumber syntax element or need to rely on the error resiliency of the compression scheme used.

The timestamp shall convey a reading of a local clock at the time the RTP packet is sent.

An SSRC value for each RTP session shall be selected randomly, as described in RFC1889 [5].

2.3.2 RTP Payload

The RTP payload consists of one or more complete FlexMux-PDUs as visualized in the figure below. Each FlexMux-PDU consists of an index element that identifies the content of the FlexMux-PDU, the length of the payload, a version number (only for MuxCode mode) and the payload itself as specified in [1]. FlexMux-PDUs are byte-aligned in the payload of the RTP packet.

Note: It may be computationally beneficial to word-align the FlexMux-PDUs. However, on average this means a 2 byte per FlexMux-PDU overhead for alignment. That seems to be rather high.

```
  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         |  RTP
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                           timestamp                           |  Header
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           synchronization source (SSRC) identifier            |
 +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 :            contributing source (CSRC) identifiers             :
 :                             ....                              :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    index      |   length      | version       :FM-PDU Payload |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...............+               |  RTP
 |     ......        .......          ........      .......      |
 |               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  Payload
 |     ......    |     index     |    length     |      .......  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
 :     ......      FM-PDU Payload    ........      .........     :
 :                                                               :
 |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     ......      ........      :...........RTP padding (opt.)  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The operation of the RTP unpacker for this payload type can be visualized by the figure below. Note that an implementation may as well integrate RTP parsing and FlexDemulitplexing in one unit.

```
  Natural/Synthetic Audio & Visual data,
     Scene Description & OD streams

  ^     ^     ^                      ^
  |     |     |                      |
  |     |     |                      |
+---+ +---+ +---+                  +---+
|AL | |AL | |AL |. . . . . .       |AL |
+---+ +---+ +---+                  +---+
  ^    ^      ^                      ^
  |    |      |                      |
==|====|======|======================|===DAI===
  |    |      |                      |
+---------------------------------------+
|     FlexMux demultiplexer             |
+---------------------------------------+
        RTP packet ^
        payload    | includes signaling
                   | of packet boundaries
       +------------------------+
       |      RTP Parser        |
       +------------------------+
                   ^
                   I
                   I
                   I
```

A Authors' Addresses

Carsten Herpel
THOMSON multimedia
Goettinger Chaussee 76
30453 Hannover
Germany
Email: herpelc@thmulti.com

Vahe Balabanian
Nortel
P.O.Box 3511, St. C
Ottawa, Ontario
Canada K1Y 4H7
Email: balabani@nortel.ca

Andrea Basso
AT&T Labs - Research
100 Schultz Drive,
Red Bank, NJ 07701
USA
Email: basso@research.att.com

M. Reha Civanlar
AT&T Labs - Research
100 Schultz Drive,
Red Bank, NJ 07701
USA
Email: civanlar@research.att.com

Don Hoffman
Sun Microsystems, Inc
901 San Antonio Road
Palo Alto, CA 94303-4900
USA
Email: don.hoffman@eng.sun.com

Michael F. Speer
Sun Microsystems, Inc
901 San Antonio Road
Palo Alto, CA 94303-4900
USA
Email: michael.speer@eng.sun.com

Henning Schulzrinne
Columbia University
Dept. of Computer Science
1214 Amsterdam Avenue
New York, NY 10027
USA
Email: schulzrinne@cs.columbia.edu

B Bibliography

[1] ISO/IEC 14496-1 CD ‘MPEG-4 Systems’ Oct. 1997

[2] ISO/IEC 14496-2 CD ‘MPEG-4 Visual’ Oct. 1997

[3] ISO/IEC 14496-3 CD ‘MPEG-4 Audio’ Oct. 1997

[4] ISO/IEC 14496-6 CD ‘Delivery Multimedia Integration Framework’, Oct. 1997

[5] Schulzrinne, Casner, Frederick, Jacobson ‘RTP: A Transport Protocol for Real Time Applications’ RFC 1889, Internet Engineering Task Force, Jan. 1996.

[6] S. Casner, V. Jacobsen ‘Compressing IP/UDP/RTP Headers for Low-Speed Serial Links’, draft-ietf-avt-crtp-04, Internet Engineering Task Force, Nov. 1997

[7] M. Handley, V. Jacobson ‘SDP: Session Description Protocol’, Internet Engineering Task Force, Nov. 1997

[8] Handley, Schulzrinne, Schooler ‘SIP: Session Initiation Protocol’, Internet Engineering Task Force, Nov. 1997

[9] H. Schulzrinne, A. Rao, R. Lamphier ‘Real Time Streaming Protocol (RTSP)’, Internet Engineering Task Force, Nov. 1997