Transport of MPEG-4 over IP/RTP

Andrea Basso
AT&T Labs – Research, 200 Laurel Ave., Middletown, NJ 07748, USA
basso@research.att.com

Socrates Varakliotis*
Computer Science Dept., University College London, Gower Street, London WC1E 6BT, UK
S.Varakliotis@cs.ucl.ac.uk

Roberto Castagno
Nokia Mobile Phones, Visiokatu 1, 33720 Tampere, Finland
Roberto.Castagno@nokia.com

Florin Lohan
Digital Media Institute, Tampere Univ. of Technology, Tampere, Finland
lohanf@cs.tut.fi

Abstract. In this paper we review and discuss some of the current efforts and proposals in the standardization bodies IETF and MPEG for the delivery of MPEG-4 over IP. We address the issues of payload format and media control by describing two architectures that allow for MPEG-4 streaming. Finally, we describe some MPEG-4 over IP streaming experiments in which the RTP MPEG-4 payload format is used in conjunction with per-stream Forward Error Correction support, in presence of packet loss.

1Introduction*

MPEG-4 is an ISO standard that brings together three key fields: digital television, interactive graphics applications (synthetic content) and the World Wide Web (distribution of and access to content). It provides a standardized and worldwide-accepted technological framework that enables the integration of the production, distribution and content access paradigms. MPEG-4 leverages existing digital video content by allowing the use of MPEG-1 and MPEG-2. Furthermore, it enables richer development of new digital video applications and services.
An MPEG-4 scene is composed by a number of audio-visual objects, which can be of either static or time-varying nature. The data generated by the encoders and relative to each of the audio-visual objects are transported by means of one or more Elementary Streams (ES) 6.
The system part of the MPEG-4 standard deals with the description of how the Elementary Streams are related to each other at different levels so as to constitute the audio-visual scene.
In particular, one can identify two main levels of description 6. The *structural* level describes how the different audio-visual objects are organized into the audio-visual scene. The objects are described and are attributed as position in the two- or three-dimensional space. The user can interact with the audio-visual objects, for example by changing his/her own point of view in a virtual scene, or by changing some properties of the objects themselves, e.g. the color or the acoustic response of an environment.
To achieve this functionality, MPEG-4 provides the Binary Format for Scenes, a language that allows the description and the dynamic evolution of the scenes. BIFS is a close relative of the virtual Reality Description Language (VRML), the main difference being that VRML is a textual description, whereas BIFS is in a binary, more compact, format.
In the system perspective, BIFS data are organized in Elementary Streams (ESs), which are transported in a coordinated manner together with the other data components.
The second description level goes into the detail of how the single ESs relate to each other in constituting the audio-visual objects. Issues such as synchronization, decoder configuration, location of the streams, as well as intellectual property and additional information relative to the single streams are handled at this level, also known as *media object description framework.*
This framework is organized as a hierarchy of descriptors that refer to each other and to external resources. The *Object Descriptors (OD)* lists all components that constitute a single media object (e.g. a video clip, a talking head, or

* This work has been partially completed and supported by an AT&T Labs – Research internship program.

a sound effect). Each such component is described, inside the OD, by means of an *Elementary Stream Descriptor*, briefly described below. As it is the case for the BIFS data, also the ODs are conveyed by means of dedicated ESs.

A particular OD is the *Initial Object Descriptor (IOD)*, which contains the Elementary Stream Descriptors relative to the ESs that carry the data relative to BIFS and the other ODs. In turn, the *Elementary Stream Descriptors* identify each stream by means of an Elementary Stream Identifier (ES_ID) or, alternatively, by a URL. A number of sub-descriptors contained in the ES Descriptors convey information specific to the stream, such as decoder configuration parameters, intellectual property information, quality of service requirements.

The richness of the content structure that MPEG-4 addresses is quite challenging when it comes to devising an appropriate transport strategy. An MPEG-4 presentation might involve a large number of streams located in various servers with different QoS and timing requirements.

The Delivery Multimedia Integration Framework (DMIF) 6 is a layer whose purpose is to de-couple the delivery from the application and to ensure end-to-end signaling and transport interoperability between end-systems 6. The interface between the application and the delivery layer is called DMIF Application Interface (DAI).

The transport of MPEG-4 content over the Internet is articulated in the media transport and the media control components. The media transport has been addressed in a joint effort between the Internet Engineering Task Force (IETF) and the Systems Group of MPEG, resulting in the publication of Internet Drafts that address the definition of payload formats for the transport of MPEG-4 content over the Internet. The control-related issues are currently part of an ongoing discussion between the MPEG System Group and the IETF.

Some implementations of the MPEG-4 client-server architecture follow entirely the DMIF model and apply the full DMIF signaling. However, the existence of the DAI allows also hybrid architectures, in which the DAI primitives are mapped on functions provided by non MPEG-specific protocols

A preliminary proposal for the mapping of DAI primitives onto the Real Time Streaming Protocols 6 is presented in Sec. 5. Its implementation is described in Sec. 5.3.

2MPEG-4 MEDIA TRANSPORT

In the MPEG-4 architecture, the elementary streams (ESs) generated from the media encoders, are passed to the Sync Layer (SL), together with information relative to boundaries of media Access Units (AUs), random access points (RAP), as well as the time at which the AUs should be composed at the receiver side (Composition Time Stamps or CTS). The Sync Layer outputs SL packets, each containing a header, which encodes information conveyed through the Elementary Stream Interface (ESI). An AU that is larger than an SL packet gets fragmented. Subsequent SL packets containing remaining parts of the AU are generated with subset headers until the complete AU is fully packetized. The syntax of the Sync Layer is flexible and allows for the presence or absence of individual syntax header elements, as well as variable field length.

2.1PROTOCOL ALTERNATIVES

Given the support provided in the SL for several transport functions such as time-stamping and sequence numbering, one can think that the most suitable way to deliver MPEG-4 over IP is to encapsulate the SL packets directly in IP/UDP packets. On the other side, a very large part, if not all, of the media carried over IP are and will be carried over RTP. Thus, transporting MPEG-4 over IP without RTP support would make interaction and integration with non MPEG-4 media streams difficult. Furthermore, such choice would make MPEG-4 a “closed” system: as an example, without RTP support, the MPEG-4 dynamic scene and session control concepts cannot be extended to non MPEG-4 data.

If RTP is then the protocol of choice for the delivery of MPEG-4 over IP still the problem on how to encapsulate the large variety of media that MPEG-4 covers need to be addressed. The encapsulation of MPEG-4 ESs over RTP by means of individual payloads is the most suitable alternative for coordination in accordance of the Application Level Framing (ALF) approach. Its complete implementation requires the definition of a large number of payload types and might lead to constructing new session and scene description mechanisms. Considering the potential effort required, which essentially reconstructs MPEG-4 systems, this may only be a long term alternative if no other solution can be found. Furthermore the Elementary Stream Interface is informative and not normative in the current specification thus one cannot expect to have access to the MPEG-4 ESs in all the MPEG-4 implementations.

A simpler and more directly implementable alternative to the “one different payload for each different media” approach is to define a single MPEG-4 RTP payload format, which carries SL-packets instead of raw ESs. Furthermore the fact that the MPEG-4 ESs provide (or are supposed to provide) support for error resilience reduces the need for error resilience support at the RTP level.

It is reasonable to minimize the redundancy, where possible, between the RTP and the SL headers. In the current specification 6, the RTP header is followed by a reduced SL header. The relevant pieces of information for the transport and synchronization are available at the RTP header level. The MPEG-4 SL specific information is contained in the reduced SL-header inside the payload.

This approach offers the advantage of simplicity and proper interface to the existing normative MPEG-4 specification, but it does not offer specific support for error resilience. It relies on generic error protection mechanisms such RTP Forward Error Correction 6 or RTP packet interleaving. In the next sections we will show some results on the performances of this payload format when associated with RTP FEC.

2.2The MPEG-4 SL payload format

The MPEG-4 SL RTP Payload 6 carries a single SL packet, including a reduced SL packet header with the SL sequenceNumber and SL compositionTimeStamp fields removed. The reduced SL Header must be zero padded to byte-align the SL packet payload. The size of the SL packets should be adjusted such that the resulting RTP packet is not larger than the path-MTU. In the following, we describe the meaning of the most important payload fields. For a more extensive description the reader is referred to 6.

RTP Sequence Number: The RTP sequence number (RTP-sn) is derived from the sequence number field of the SL packet. A random offset can be added for security reasons 6. While the RTP sequence number is a fixed 16-bit field the SL sequence number (SL-sn) can be variable in size. If the latter has less than 16-bit length, the unused MSBs of the RTP-sn are initially filled with a random value that is incremented by one each time the SL-sn wraps up. In case of SL-packet duplication, indicated by a null SL-sn in multiple consecutive SL packets, the RTP sequence number is incremented by one for each of these packets after the first one. The RTP packetizer will generate its own RTP-sn if no SL-sn field is present in the SL packet header.

RTP Timestamp: The RTP timestamp is set to the value of the CTS. The SL Composition Time Stamp (CTS) has a variable resolution and can be optional in contrast with the RTP timestamp that is mandatory and 32 bits in length. If CTS has less than 32 bits in length, the MSBs of the RTP timestamp must be set to zero. In case the CTS is not present in the current SL packet, but has been present in a previous SL packet, this same value must be taken again as RTP timestamp. If CTS is never present in SL packets for a given stream, the RTP packetizer should use as RTP timestamp a reading of a local clock at the time the RTP packet is created. The resolution of such timestamp should be conveyed explicitly through some out-of-band means.

RTP Timestamp in RTCP SR packets: According to the RTP timing model, the RTP timestamp that is carried into an RTCP SR packet is the same as the CTS that would be applied to an RTP packet for data that was sampled at the instant the SR packet is being generated and sent. The RTP timestamp value is calculated from the NTP timestamp for the current time, which also goes in the RTCP SR packet. To perform that calculation, an implementation needs to periodically establish a correspondence between the CTS value of a data packet and the NTP time at which that data was sampled.

RTP SSRC: set as described in RFC1889 6. A mapping between the ES identifiers (ES_IDs) and SSRCs should be provided through out-of-band means. CC and CSRC fields are used as described in RFC 1889 6.

3Implementation

The system we have developed has three components, as depicted in Figure 1. An RTP/RTCP/RTSP server that has been extended with a back-end to serve MPEG-4 content, an RTP/RTCP/RTSP client that has been integrated with the IM-1 MPEG-4 player version 3.5, and a packet loss generator that is capable of introducing a pre-specified user programmable percentage of loss.

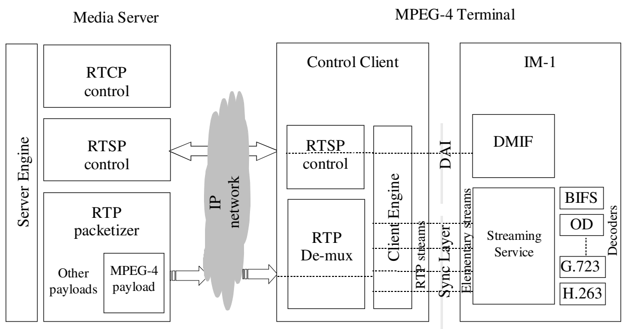

Figure 1: MPEG-4 streaming architecture.

3.1Server

The server reads a stored MPEG-4 presentation in TRIF format (the one supported in version 3.5 of the IM1 player 6). That was the only stable MPEG-4 format and IM-1 version available at the time of the implementation. An MPEG-4 file format library has been made available in source code very recently.

The information in the DecoderConfigDescriptor maxBitrate, avgBitrate and StreamType fields is used to configure properly the server. Critical data such as IODs are transferred via TCP at the beginning of the MPEG-4 presentation. Media control is set up and handled via RTSP messages between server and client**.** The server streams every component of the MPEG-4 presentation as a separate RTP session 6. According to the RTP specification every RTP session is identified by a different SSRC. The client requests every stream component on demand from the IM-1 player via DMIF calls that are implicitly mapped into RTSP calls via the IM-1 file interface. Section 5 will describe and explicit mapping DMIF-RTSP.

The MPEG-4 streams are packetized with a single SL-Packet per RTP payload. Due to the variable size of the SL sequence number and SL timestamp and the fact that bit-parsing of the SLConfigDescriptor is needed to determine their length, the processes of SL-packet and RTP generation has been coupled at the server side, for efficiency reasons to avoid extra bit-shifting.

3.2Client

The client is composed of two components: the RTP/RTSP client and the IM-1 MPEG-4 player v3.5 6. A DMIF plug-in has been implemented to support RTP/RTSP additions in IM1, without affecting the core part of the IM-1 player. The RTP/RTSP client connects to the DMIF plug-in of IM-1 player via a TCP local socket. Ring buffering is performed at the DMIF plug-in. A presentation is started as soon as the RTSP URL is entered via the IM-1 user interface. By means of the new DMIF instance, the URL is passed via a local socket to the RTP/RTSP client that takes care of setting up the streams and issues a per-stream RTSP PLAY command. Loss monitoring is done at the client side by detecting discontinuities on the sequence number and the timestamp. MPEG-4 client implementations that do not produce full SL packets, but rather produce the RTP header and the reduced SL header and pass these to the MPEG-4 system decoder are preferable and more efficient. They do not require, in fact, the knowledge of the SLConfigDescriptor and the evaluation of the compositionTimeStampFlag at RTP depacketizer level. However, the choice between generating SL packets and converting or generating RTP directly is an implementation detail.

3.3MPEG-4 content

An in-house MPEG-4 presentation called *BoldyTalk,* has been used in the tests. It is composed of a G.723.1 audio stream coded at 6.3 Kb/s, an H.263 video stream coded at 128 Kb/s and 15 fps, a JPEG image that constitutes the background, an IOD stream, and a BIF stream. The OCR stream is coupled with the audio stream. The JPEG image is packetized in 11 SL-packets of 1450 bytes in size. The audio is packetized in 2350 SL-packet of 20 bytes each in length. The H.263 video resulted in 1632 SL-Packets of variable size between 2 and 1470 bytes. The path-MTU was 1500 bytes. The SL-Packet length is limited to 1450 bytes. The MPEG-4 presentation is organized in two RTP/RTCP streams as shown in Table 1. The RTP packetization is performed on the fly. The packetization CPU time is less that 1%. We observed that on average the SL-reduced header is 1 byte in size. The numbers of packets for which the SL-header is 2 of 4 bytes is small (the first packet of every H.263 video frame has a reduced SL-Packet header of 2 bytes, and the first packet of the G.732.1 has a reduced SL-Packet header of 4 bytes).

Table 1: MPEG-4 Stream Configuration.

Stream 1	RTP JPEG + G.723.1 audio + OCR RTCP JPEG image + G.723.1 + OCR
Stream 2	RTP H.263 stream RTCP H.263 stream

4Experimental results in presence of packet losses

We ran some preliminary tests to evaluate the performances of the system described in Figure 1 in presence of losses. We used the example BaldyTalk for the experiments. Furthermore we included RTP per-stream Forward Error Correction support [3].

4.1RTP FEC payload format

Forward Error Correction (FEC) is a technique that has been proposed in the literature to compensate for packet loss. IETF is in the process of standardizing a payload format for generic forward error correction of media encapsulated in RTP 6. It is engineered for FEC algorithms based on the exclusive-or (parity) operation and allows the use of arbitrary block lengths and parity schemes. Such payload allows for the recovery of the media payload it is associated with, as well as critical RTP header fields of the RTP media stream. In our implementation the FEC data is computed on a per-stream basis. It is then sent as a separate RTP stream and associated with the corresponding RTP media stream via SDP.

The forward error correction is implemented in the following way: The sender takes some set of packets from the media stream, and applies an XOR operation across the payloads. The sender also applies the XOR operation over components of the RTP headers. Based on the procedures defined in 6, the result is an RTP packet containing FEC information. This packet can be used at the receiver to recover any one of the packets used to generate the FEC packet. Furthermore, each FEC packet contains a 24-bit field called the offset mask and a 16-bit field, N, called the sequence number base. If bit i in the mask is set to 1, the media packet with sequence number N + i was used to generate this FEC packet. The offset mask and payload type are sufficient to signal arbitrary parity based forward error correction schemes with little overhead. The reader is referred to 6 for further details. In the implementation the FEC streams are computed on the fly, during media delivery.

4.2Packet loss generator

This module was developed to introduce user-specified percentage of packet losses to the RTP-packetized MPEG-4 streams. Earlier literature indicated that individual packet loss models could be used to simulate loss in a packet network 6 such as the Internet. Recent literature 66 suggests that loss patterns for applications such as packet video and audio, as more suitably modeled as bursty. In our implementation relied on the well-known Gilbert model shown in Figure 2.

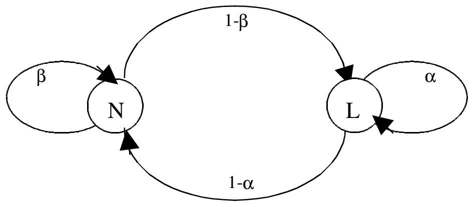

Figure 2: Gilbert model for packet loss simulation.

State N represents no losses, whereas L is a lossy state. The probabilities of being in one of the states, L or N, are denoted as α and β respectively. Then, the matrix P:

$$P = \begin{bmatrix} \beta & 1-\beta \\ 1-\alpha & \alpha \end{bmatrix}$$

represents the model transition probabilities. The model is completely defined by two parameters, the transition probability from state N to state L (1- in Figure 2) and the probability of being in L called also unconditional loss probability or p_{ul}. Note that 1- can be computed independently from the model only on the basis of p_{ul} and p_{cl}, where p_{cl} is the conditional probability of loss:

$$\beta = 1 - p_{ul}\frac{1-p_{cl}}{1-p_{ul}} = 1 - p_{ul}\frac{1-\alpha}{1-p_{ul}} \qquad (1)$$

Note also that the Gilbert model implies a geometric distribution for remaining in state L. Let's call *b* this average burst length, or the 'expectation' of a burst to occur. Then, let's assume that we are in state L in the model of Figure 2. The probability of loss of a single packet is:

$$P(b = 1) = 1 - \alpha$$

while the loss of n consecutive packets can be expressed as:

$$P(b = n) = \alpha^{n-1}(1-\alpha)$$

Then, the average burst length *b* is related to α:

$$b = \sum n\alpha^{n-1}(1-\alpha) = \frac{1}{1-a} \Rightarrow \alpha = 1 - \frac{1}{b} \qquad (2)$$

Note that the average burst length depends only on the conditional loss probability, i.e. on the loss behavior between two consecutive packets.

By means of equations 1 and 2, given an average burst length and the unconditional loss probability, the model of Figure 2 is completely defined.

In 6 Bolot investigated the question of the degree to which packet losses can be well modeled as independent. The relationship between the unconditional loss probability p_{ul}, and the conditional loss probability p_{cl}, based on the condition that p_{cl} applies if the previous packet was lost is discussed. It is concluded that for different packet spacing (intervals between 8 ms and 500 ms are considered), p_{cl} approaches p_{ul} as the packet spacing increases. This indicates that a bursty loss environment is characterized by the presence of short-lived loss correlation, or short average loss length. We further investigated the remarks presented in 6 according to the above analysis 6.

4.3Experimental results

We performed a series of experiments with our MPEG-4 streaming testbed, as described in 6, 6. The results, for various FEC lengths, are shown in Table 2.

Table 2: Percentage of recovered packets for the complete MPEG-4 presentation.

Loss rate (%) :		0.1	0.5	1	3	10
FEC length	Overhead					
3	33%	100	100	100	96	86
7	14%	100	100	90	85	50
15	7%	100	100	89	70	21

Table 2 shows the average percentage of recovered packets for the complete MPEG-4 presentation. Experiments showed that FEC has good performances when the loss rate is of the order of few %. For higher loss rates a more sophisticated FEC scheme or a combination of parity FEC and other error concealment techniques may be necessary.

5MPEG-4 streaming media control based on DMIF-RTSP mapping

This implementation is based on an original mapping 6 of MPEG specific signaling primitives onto RTSP 6 (Real Time Streaming Protocol). The architecture is shown in Figure 3.

5.1Server

The server used for our experiments has been obtained as a modification of the *mux.exe* program distributed in the framework of MPEG. The simple structure of the server does not allow for complex functionalities (e.g. server interaction). However, this initial tool allowed us to achieve the objective of streaming over the Internet a simple audio-visual presentation. During the 49th MPEG meeting in Melbourne, the interoperability of the media transport of a previous version of this server (developed by Nokia) with the client developed by AT&T has been demonstrated 6.

The server receives in input the elementary streams (Object Descriptor, BIFS streams and media streams), just as the *mux.exe* program does, creates the Sync Layer packets and maps them onto RTP packets.

5.2 Client

The client consists of a DMIF-compliant plug-in to the IM-1 (with a small exception mentioned later). The plug-in parses the RTP packets, reconstructs the SL packets and sends them to the Sync Layer. Decoding and rendering complete the process. In order to display the content of the presentations in our simulations, we used the version 3.5 of the publicly available "2D player".

5.3 Implementation details

The media transport of this architecture is based on the payload format that has been jointly proposed by the MPEG System Group and the IETF 6. As far as the media control (shaded in darker gray in Figure 3) is concerned, the proposed system is based on the idea of mapping the primitives required at the DAI interface onto a subset of RTSP functions. This appoach allows us on the one hand to maintain the compliance to the DMIF philosophy by decoupling the application from the network details, and on the other hand to achieve interoperability with generic servers that operate using RTSP. As mentioned in section 5, such mapping is not a trivial issue, and discussion on this topic is ongoing in both MPEG and IETF.

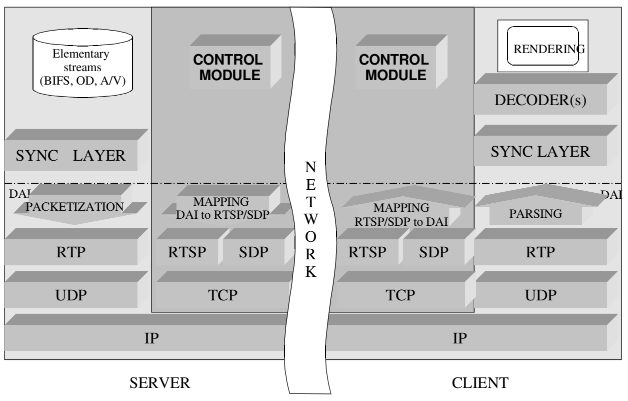

Figure 3: General structure of the Nokia architecture.

In the following sections, we shall describe step by step the phases that characterize a session of MPEG-4 networking.

5.3.1 Service Initialization: ServiceAttach

First, the client is establishing a session with the server by requesting a ServiceAttach on its side. The client is providing the URL of the desired MPEG-4 application. The DMIF layer connects to server and a new session is created on the server side. If the response to the client's request is OK, the server is sending also the IOD of the MPEG-4 application requested by the client. DMIF also has to solve at this point the correspondence between ES_IDs that identify the MPEG-4 streams at the application level and CH_IDs (channel id) that identify the network streams at DMIF level.

In our implementation we mapped the ServiceAttach command over two RTSP commands, as depicted in Figure 4.

First the client issues a DESCRIBE RTSP command. If the specified URL exists on the server side, the RTSP layer transmits the information about the correspondence between ES_IDs and ch_id-s as an Application/SDP attachment to the RTSP response. In the SDP message we describe each stream involved in the requested MPEG-4 application as an "`application`" SDP media. Additional information as the transport protocol (RTP), profile (AVP) are carried in the media announcement line. We use a separate dynamic media payload type for each stream. In order to specify the ES_ID, we use the following SDP attribute. For example, if the ES_ID for the currently described stream is 23, it would be specified by the SDP line "`a=x-mp4-esid: 23`". The ch_id is specified as a last element in the URL path. For instance, a ch_id of 5 is specified by the SDP line:

```
a=control:rtsp://server.com/movie/5
```

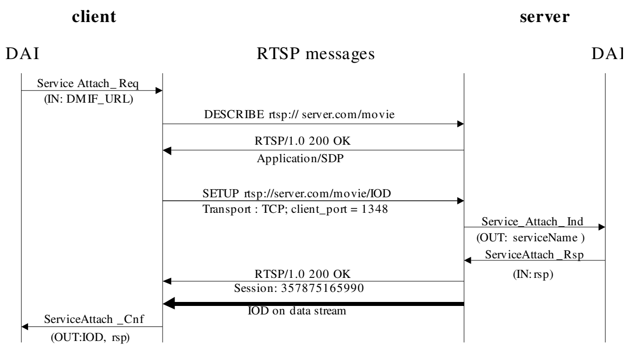

Figure 4: ServiceAttach.

After the client has interpreted this information, it issues a SETUP RTSP command, in which it requests the IOD. The URL for the command is the one specified in the ServiceAttach request, but modified by adding "`/IOD`" in the end. The server will recognize this SETUP command as part of a DMIF ServiceAttach request. As a result of this SETUP command, the server is establishing an RTSP session, issuing a Session number in its response. It will also open a data channel (TCP) and the IOD is sent to the client through this data channel. After the client has both the RTSP session number and the IOD, it calls the ServiceAttach_Cnf call-back function on the client side.

It should be observed that the RTSP session number is issued here in correspondence to the opening of the IOD channel . In this way, we practically establish a correspondence between the DMIF session and the existence of the IOD channel. The DAI serviceDetach is then mapped (see Sec. 5.3.5) on the TEARDOWN of the IOD channel. This choice is suggested by the fact that -as far as DMIF is concerned- it should be possible to close all the channel of a session and open new ones, without the session being lost. However, this is not the case with RTSP session, in that the session is closed when the last channel is TEARDOWNed.

This solution allows us to open and close (possibly also all of them) freely OD, BIFS and data channels without the RTSP session being lost.

5.3.2Opening The Channels: ChannelAdd

After the client application has opened the session and it has the IOD, it can open channels. This is made with the DMIF command ChannelAdd. Several channels can be opened with a single ChannelAdd command, that takes as parameter an array (loop) of channel description elements. A channel description element contains information like channel ID (ch_id), channel direction (downstream, upstream or bi-directional), other QoS information.

In the proposed implementation ChannelAdd was mapped over the SETUP RTSP command (see Figure 5).

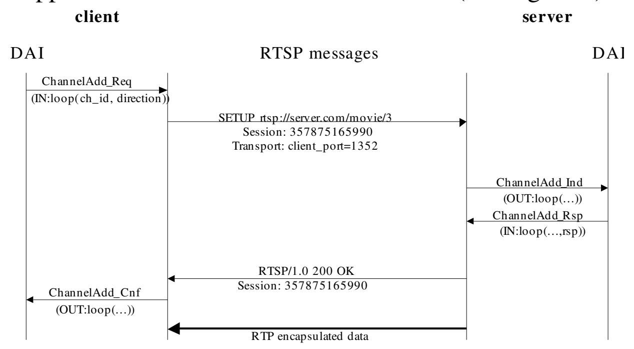

Figure 5: ChannelAdd.

A separate SETUP command is issued for each channel in the array (loop). The ch_id is specified in the URL as described above. The transport RTSP field is specifying the stream direction, as other transport parameters. If the command is successful, a data stream is created, the server is going to start the RTP streaming. For the video stream, for example, the succession of communications is:

The direction of the channel is specified by the "`mode`" parameter of the Transport field. DMIF specifies 3 types of channels: downstream, upstream and bi-directional. A downstream direction is mapped to "`mode=play`", an upstream direction is mapped to "`mode=record`" and a bi-directional channel is specified as "`mode=play, record`".

Also an important aspect of the mapping is the handling of the ChannelAdd commands on client and server sides. If the client issues a ChannelAdd request with a loop containing more than 1 channel, a SETUP RTSP command is issued separately for each channel. On the server side for each received SETUP command a ChannelAdd is issued with a loop containing 1 channel. The reason for this is that RTSP does not allow issuing a new command before receiving an answer for the last issued command.

5.3.3Controlling the Channels: UserCommand

After the client application has opened channels, it can issue commands on those channels.

DMIF only defines a generic user command, since there can not be a priori knowledge about specific commands needed by an application. Because our application only streams data and we do not aim at complex interactive commands, our implementation only supports playing and pausing as user commands. An array (loop) of user commands is provided to the UserCommand function.

We mapped these two commands over the PLAY and PAUSE RTSP commands (one for each channel), as depicted in Figure 6. The play command also carries a range and a speed, as RTSP fields. The pause command can only carry a range.

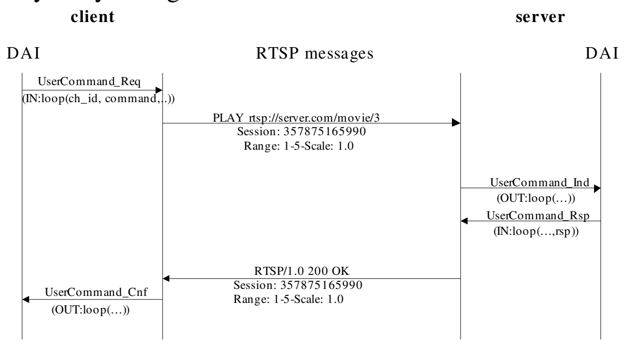

Figure 6: UserCommand.

An additional RTSP extension field called DMIF-Acknowledge allows the server to make the difference between a DAI_UserCommand_Req and a DAI_UserCommand_Ack_Req on the client side. The RTSP response is sent anyway (as stated in the RTSP standard) but if the DMIF-Acknowledge field is "No", then the DAI_UserCommand_Ind is called on the server side instead of DAI_UserCommand_Ack_Ind.

5.3.4Closing the Channels: ChannelDelete

After it finishes playing, the client application can request closing of channels.

An array (loop) of channel deleting description is provided to the ChannelDelete function. The client application is specifying the ch_id and a reason for deleting the channel.

We mapped the ChannelDelete command over the TEARDOWN RTSP command (Figure 7). There is one TEARDOWN command for each channel delete description, containing the ch_id in the URL as described above and the deleting reason as RTSP extension field.

As for the ChannelAdd command, on the server side one received TEARDOWN command triggers a DAI_ChannelDelete_Ind callback.

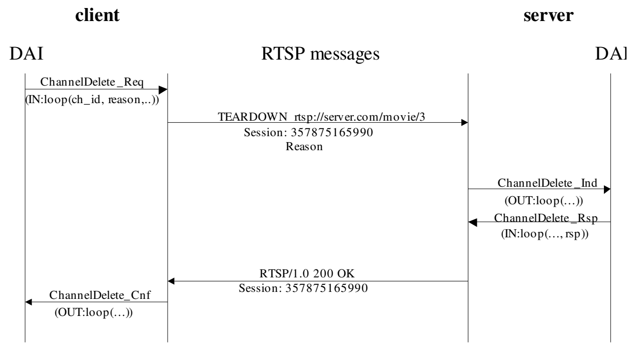

Figure 7: ChannelDelete.

5.3.5Terminating the Session: ServiceDetach

After the presentation is over, the client can choose to close the session and to free all the network resource using the ServiceDetach command.

We mapped the ServiceDetach command over the TEARDOWN RTSP command (Figure 8). The RTSP stream that is closed by this command in the one for the IOD that was opened with the SETUP command in the ServiceAttach command. Our implementation also teardowns all the RTSP channels that are still open before issuing the TEARDOWN for the IOD channel.

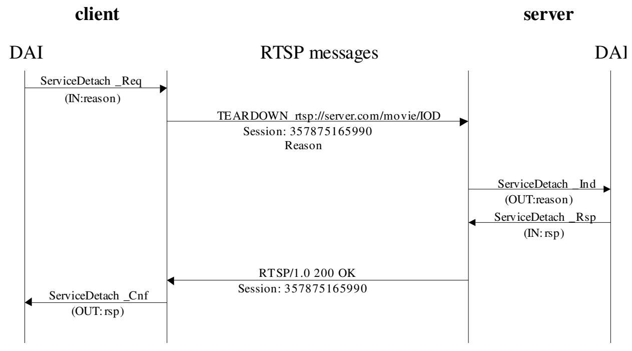

Figure 8: ServiceDetach.

6Conclusions

In this document, after a review of the current efforts in standardization of MPEG-4 over IP, we have described some experiments that we ran in order evaluate the performances of FEC in the delivery of MPEG-4 SL-packetized streams in presence of packet losses. Results show that parity FEC can offer a good error correction support in MPEG-4 streaming of SL-packetized streams when the loss rate is in the range of few %. Furthermore, an architecture for the media control based on the mapping of DAI to RTSP has been presented.

Acknowledgment

The authors wish to thank Lèon Bottou from AT&T Labs for the discussions on the package loss generator.

Manuscript received on …

References

[1] R. Civanlar, A. Basso, S. Casner, C. Herpel, C. Perkins, RTP Payload Format for MPEG-4 Streams. IETF Internet Draft, March 2000.

[2] ISO/IEC/SC29/WG11. Generic Coding of Moving Pictures and Associated Audio (MPEG-4 Systems CD), ISO/IEC 14386-1. International Standards Organization, 1999.

[3] J. Rosenberg, H. Schulzrinne. An RTP Payload Format for Generic Forward Error Correction. RFC 2733, Internet Engineering Task Force, June 1999.

[4] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real Time Applications. RFC 1889, Internet Engineering Task Force, January 1996.

[5] ISO/IEC/SC29/WG11. FDIS Delivery Multimedia Integration Framework, ISO/IEC 14496-6. International Standards Organization, November 1998.

[6] A. Basso et al. Preliminary results in Streaming MPEG-4 over IP with the MPEG-4/IETF-AVT payload format. ISO/IEC/SC29/WG11 m5027, Melbourne, July 1999.

[7] A. Basso, S. Varakliotis. Per-stream FEC performances in streaming MPEG-4 over IP. ISO/IEC/SC29/WG11 m5479, Maui, USA, November 1999.

[8] . Basso, S. Varakliotis. Transport of MPEG-4 over IP/RTP. In *IEEE International Conference on Multimedia (ICME 2000)*, New York, USA, August 2000.

[9] C. Herpel. Elementary Streams Management. In *IEEE Transactions on Circuits and Systems of Video Technology,* Vol. 9, No. 2, March 1999.

[10] H. Kalva, A. Eleftheriadis. Delivering MPEG-4 content. In *Packet Video Workshop*, New York, USA, April 1999.

[11] R. Castagno et al. MPEG-4 over IP: preliminary results using MPEG/IETF AVT payload and DAI/RTSP mapping. ISO/IEC/SC29/WG11 m5554, Maui, USA, November 1999.

[12] R. Castagno, S. Kiranyaz, F. Lohan, I. Defée. An architecture based on IETF protocols for the transport of MPEG-4 content over the Internet. In *IEEE International Conference on Multimedia (ICME 2000)*, New York, USA, August 2000.

[13] H. Schulzrinne, A. Rao, R. Lanphier. Real-Time Streaming Protocol (RTSP). RFC 2326, Internet Engineering Task Force, April 1998.

[14] J-C. Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *SIGCOMM '93*, pages 289-298, September. 1993.

[15] V. Paxson. End-to-End Internet Packet Dynamics. In *SIGCOMM '97*.

[16] I. Cidon, A. Khamisy, M. Sidi. Analysis of packet loss processes in high-speed networks. In *IEEE Transactions on Information Theory*, Vol. 39, No. 1, pages 98-108, January 1993.

[17] http://mpeg4.nist.gov/IM1/downloads.html