RTP PACKETIZATION OF MPEG-4 ELEMENTARY STREAMS

Matthias Ohlenroth and Hermann Hellwagner

Department of Information Technology
University Klagenfurt
Universitätsstraße 65-67
A-9020 Klagenfurt
Austria

ABSTRACT

Multimedia streaming is becoming increasingly popular. The multimedia standard MPEG-4 was designed to support scenes of different levels of complexity and applications with low bandwidth up to very high bandwidth requirements. Scalable video encoding is supported as well. This allows MPEG-4 streaming software to adapt video quality dynamically in the network to currently given QoS conditions. We evaluate packetization modes designed to transport MPEG-4 elementary streams over RTP connections, based on an implementation of an MPEG-4 video server and a demo client. Suitability of the packetization modes for video stream adaptation in the network nodes is discussed.

1. INTRODUCTION

Multimedia presentations including digital video are becoming ever more popular as the Internet evolves. This includes live broadcasting of videos, live interactive video and the delivery of prerecorded video. Two basic delivery techniques can be used:

- download and play, and
- streaming.

In the first case, the whole video must be downloaded and stored locally. This can be a time-consuming and space-consuming process. Streaming on the other hand allows a client to start playing as soon as a sufficient portion of the video has been received. This mode minimizes the time-consuming download process prior to the playing phase and the local storage requirements. But it depends heavily on the network conditions during play-back. The streaming mode requires certain quality of service (QoS) guarantees from the network it uses. Unfortunately, the heterogeneous nature of the Internet does not allow QoS contracts to be worked out. Hence, transmission parameters like bandwidth, packet delay, and jitter depend on the current network load and vary heavily over time. Because the Internet relies in most parts on best-effort resource scheduling techniques, bursty network traffic and congestion can be observed. Two solutions are possible:

- The client-server pair uses appropriate protocols like RTP/RTCP [1] to measure network QoS parameters. Using these measurements, the server adjusts the transmission bandwidth. Consequently the perceived video quality on the client side varies over time, especially sudden or frequent video quality changes are possible. This method is called *non-transparent* video scaling as it involves both client and server [2].
- The second method shifts the video adaptation task from the server into the network because the network routers have timely knowledge about current QoS situations. This is called *transparent* video scaling because it does not involve the end nodes in the quality adaptation process [2].

Video adaptation seems promising in a number of situations. First, even in the case of heterogeneity of network equipment and connectivity, video data can be adapted and transmitted at a suitable rate. Second, variations of network traffic can be monitored and used to adapt video data instantly. Third, adaptation techniques can be used to scale video data to a wide variety of client end systems without special server support. End systems may be multimedia workstations, PDAs, or even mobile phones. Each of these devices has different requirements on the media stream because of their specific processing power, buffering capabilities, and displays. Additionally, the load on end systems may vary depending on the applications currently running.

Our research focuses on the recent multimedia transfer standard MPEG-4. MPEG-4 [3] allows the construction of audio-visual scenes consisting of separate audio and video objects. Each object is encoded separately into one or more so-called elementary streams (ESs). Scalable encoding can be used for video objects such that the decoder can produce decoded objects with different quality depending on the number of elementary streams used for decoding. The transport of elementary streams is based on the notion of access units (AUs). Access units are the smallest units to which time information can be attributed. These may be video frames.

Two different mechanisms to transfer MPEG-4 sessions over networks exist:

- The delivery multimedia integration framework (DMIF, Part 6 of MPEG-4) [4] provides a generic interface to physical transfer and storage. One implementation that is based on the UDP protocol has been developed at the University of British Columbia [5].
- An RTSP/RTP based transport mechanism is developed in a joint effort by IETF and MPEG.

To support the second method, the following specifications are standardized or under development:

- **RFC 3016.** This specification describes RTP payload formats for transport of MPEG-4 audio and visual bitstreams without using MPEG-4 Systems streams.

- **Draft-ietf-avt-mpeg4-multiSL-04.txt.** This payload format specification may be used to transport any MPEG-4 elementary stream with or without MPEG-4 Systems streams.
- **Draft-ietf-avt-mpeg4-simple-02.txt.** This payload format specification is a simplified version of the previous one. Therefore it is not discussed here.
- **Draft-curet-avt-rtp-mpeg4-flexmux-02.txt.** This specification allows the transport of FlexMux streams over RTP. FlexMux streams are interleaved elementary streams.

Utilizing RTP-based transport, we are developing an end-to-end MPEG-4 video streaming solution that incorporates nodes capable of scaling video streams during transmission. Therefore we investigated the mapping of MPEG-4 elementary streams onto the RTP transport protocol both conceptually and practically. The experiments done are based on a video streaming solution presented in this paper. It consists of a video server and a client, both are able to use the three mapping modes mentioned above.

The paper is organized as follows. First, the mapping methods to transport MPEG-4 data over RTP are introduced. The next section introduces our video server. The following sections compare the mapping modes both on a conceptual and on a practical level and discuss their usability in a streaming environment with adaptivity.

2. PAYLOAD SPECIFICATION RFC 3016

RFC 3016 [6] defines a payload format for carrying MPEG-4 audio and MPEG-4 visual bitstreams without using MPEG-4 Systems. It specifies the use of RTP header fields and fragmentation rules for the mapping of MPEG-4 audio and visual streams onto RTP packets. RFC 3016 does not define additional header structures because MPEG-4 streams contain error resilience information that can be used for recovering corrupt header data.

The fragmentation rule recommends not to map more than one video object plane (VOP, usually a frame) into an RTP packet. In this case, the timestamp uniquely indicates the VOP time framing. Nevertheless, concatenating multiple VOPs in an RTP packet is allowed. This helps reducing overhead in case a VOP contains only a small number of coding blocks as may occur in arbitrary shaped VOPs. RFC 3016 recommends that a single video packet is sent as a single RTP packet.

3. MULTI-SL PACKETIZATION

This payload format, defined by [7], supports two kinds of terminals:

- Terminals that implement the MPEG-4 specification including MPEG-4 Systems.
- Terminals that implement only parts of the specification. One example would be a terminal that implements only MPEG-4 Visual but not MPEG-4 Systems.

In case MPEG-4 Systems is used, the initial object descriptor (IOD) must be sent to the receiver by out-of-band means. This may be done using RTSP [8] and SDP [9].

Depending on the terminal implementation, the synchronization layer (SL) defined by MPEG-4 Systems may or may not be used. In both cases, access units or fragments thereof are mapped onto RTP packets. Compatibility with RFC 3016 can be achieved if the synchronization layer is not used. But even if MPEG-4 Systems is not implemented, fields like the decoding timestamp of the SL may be of interest for the application. Consequently the SL header is split into generally useful information and MPEG-4 Systems related data (see figure 1). These parts are called payload header and remaining SL header (RSLH). SL header fields are mapped onto RTP header fields, the payload header and the RSLH. Payload header and RSLH may consist of multiple sections if the RTP packet encapsulates multiple AUs or AU fragments.

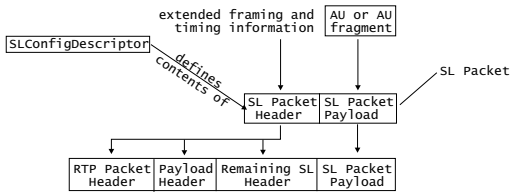

Figure 1: Mapping of AUs into SL packets and SL packets into RTP packets

4. RTP PACKETIZATION OF MPEG-4 FLEXMUX STREAMS

This payload format [10] defines the mapping of MPEG-4 FlexMux PDUs into RTP packets. The FlexMux tool supports the mapping of AUs encapsulated in SL packets of a set of elementary streams into one FlexMux stream. This mapping can be static (simple mode) or dynamic (MuxCode mode). In the latter case, configuration information must be sent to the receiver when the mapping changes. FlexMux configuration information may be transferred by out-of-band means. An integer number of FlexMux packets is mapped into one RTP packet payload. The size of FlexMux packets should be set such that the resulting RTP packet is not larger than the path-MTU.

5. RTSP/RTP BASED STREAMING VIDEO SERVER

Our goal was to implement an RTSP/RTP based MPEG-4 video server that can be used as a basis for a complete streaming environment and for experiments at the elementary stream level. The design of the server (see figure 2) is multi-threaded because it simplifies event handling and the RTP library used [11] can be decoupled from other components. Unfortunately the RTP library was not thread-safe. Hence, we had to add a wrapper layer on top of the RTP layer.

We use one thread to handle RTSP requests from multiple clients and to manage RTP connections through the session layer and the stream layer. MPEG-4 ES delivery is handled by dedicated threads. These threads read access units from the requested file, fragment them and packetize them using one of the described methods. This model simplifies the server design because handling of different streams can be decoupled and CPU utilization is managed by the operating system. An alternative solution would be to implement an event handling module that allows to schedule packet transmission requests for all streams inside the application. This would increase implementation complexity, but it could potentially reduce the overhead due to thread scheduling by the op-

erating system. With this implementation we are able to compare packetization modes.

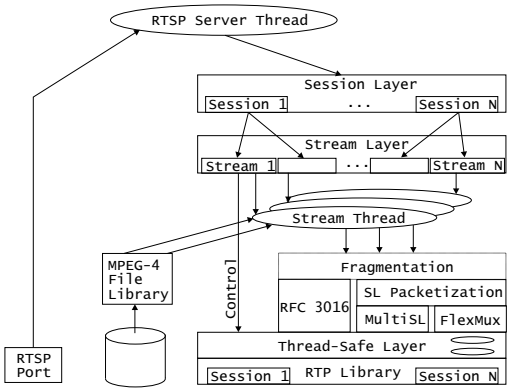

Figure 2: Design of the streaming video server

6. COMPARING PACKETIZATION MODES

To support transparent scaling, we need to carry control information that allows to steer the scaling performed by network nodes together with media data. Routers with limited processing capabilities use this information to intelligently drop packets. Control information must be accessible to these nodes without the need to decode complex protocols. More powerful nodes may perform complex transcoding tasks. Due to their processing power they are able to handle complex control information streams that describe video stream scaling properties at AU level. Such a stream may be a BSDL stream (Bitstream Syntax Description Language) [12]. Hence, a protocol is needed that allows to carry light-weight control information encoded into headers for fast access together with separate control streams and probably multiple related media streams. Furthermore, efficient access to configuration data like the decoder configuration, the SLConfigDescriptor and protocol parameters is needed by network nodes to perform transcoding.

Table 1 compares the MPEG-4 packetization modes RFC 3016, draft-ietf-avt-mpeg4-multiSL-04.txt (MultiSL) and draft-curet-avt-rtp-mpeg4-flexmux-02.txt (FlexMux) presented so far. Two modes support MPEG-4 Systems streams (BIFS stream, IPMP stream, OD stream) and one mode supports multiplexing access units from several streams into one FlexMux stream. This mechanism may be used to interleave video streams with control information streams. The RTP header extension mechanism may be used to carry light-weight control information. Unfortunately, this forces even routers to parse the RTP header because the location of this area is not fixed. The packetization modes discussed require different amounts of information to be present before RTP packets can be decoded. While RFC 3016 requires no information, the MultiSL packetization draft needs the SLConfigDescriptor and further configuration data to specify the payload header and the RSLH section. The FlexMux mode requires the StreamMappingTable (simple mode) or the MuxCodeTable (MuxCode mode) to be present. The payload modes generate different amounts of header information besides the required RTP header.

We list the size of the headers (including RTP header) for a synchronization layer containing composition time stamp, decoding time stamp (32 bits each) and a flag that signals the end of an access unit. MultiSL packetization mode uses 16 bits for the Size-Length field and 8 bits for the DTSDelta field. Using these configurations, we have measured the time to packetize and depacketize the RTP packet payload. Our tests are performed on an AMD Athlon 1 GHz CPU running the operating system Linux. Results are shown in table 1. They do not differ significantly except when fragmentation for FlexMux packetization is in use. In this case the synchronization layer for fragments following the first one is smaller and the packetization overhead reduces to 10.7 μs. RFC 3016 support has been implemented using the MultiSL code and appropriate configuration information. Hence, it might be possible to reduce the overhead for this mode if an optimized implementation is used. Implementation complexity is high for MultiSL and medium for FlexMux and can be very low for RFC 3016. Typical processing for one AU including reading from file and delivery to the network without further fragmentation takes approximately 103 μs. Hence, packetization takes approximately 20 % of the time required to process one AU.

	RFC 3016	MultiSL	FlexMux
Supported stream types	Audio, Video	Audio, Video, Systems	Audio, Video, Systems
ES to RTP mapping	Separate RTP sessions	Separate RTP sessions	Multiplexing onto one RTP session
Transport of control streams	Separate RTP session	Separate RTP session	Multiplexing onto one RTP session
Information required to decode packet	Nothing	SLConfigDescriptor and configuration parameters	StreamMappingTable, MuxCodeTable if Mux Code mode
Access to data (i.e., video packet)	Directly after RTP header	Decode payload header and RSLH section	Two bytes after RTP header, decode packet if MuxCode mode
Header overhead	12 bytes	16 bytes	15 bytes / 23 bytes
Packetization	18.0 μs	19.9 μs	19.6 μs
Depacketization	21.4 μs	23.4 μs	20.1 μs

Table 1: Features of packetization modes

7. USING PACKETIZATION MODES IN STREAMING ENVIRONMENTS WITH SCALING SUPPORT

BSDL seems to be a promising approach to transport scaling properties (i.e., metadata) closely related to video streams through the network consisting of proxy nodes and routers. To accomplish the close relationship between media data and metadata, the FlexMux packetization mode provides the required functionality. It allows access units from different streams to be multiplexed onto one RTP connection. The simple mode of the FlexMux packetization draft seems more promising than the MuxCode mode because the bandwidth allocated to different streams is not fixed and the overhead to decode FlexMux streams at router nodes is lower. Additionally, the MuxCode mode requires configuration information to be carried

by out-of-band means. In this case, a router would need special mechanisms to read and parse separate configuration connections between server and clients. Using the simple mode, one would need a configuration table that documents the (fixed) mapping between elementary stream IDs and FlexMux channels (which are identified by an index value inside a FlexMux PDU). Hence, even in this case out-of-band communication might be necessary. This situation could be changed if some of the index values were used in a special way. The range of 238 index values assigned to simple mode seems rather large for practical applications that would multiplex streams. Hence, it should be possible to use some of them to either signal a special in-band configuration stream (one stream would suffice) that can be parsed even at router nodes or to assign a set of index values to predefined stream types that must be parsed by network nodes (e.g., BSDL streams).

In the first case, the configuration stream carries packets that allow a network node to identify the type of the other streams and relations between them. Especially it should be possible to identify video streams and BSDL streams and the relations between sets of both. It is expected that one FlexMux session is assigned to a set of related video streams (e.g., base layer ES and enhancement layer ES for one MPEG-4 object) and the corresponding BSDL stream.

The second method could be used if special mappings between FlexMux index values were established. I.e., a video stream is assigned index value 0 and the corresponding BSDL stream is assigned an index value 0+128. Hence, all streams with an ID larger than 128 are identified as BSDL streams and the connection to the corresponding video stream can be established. The methods discussed here are not specific to video streams. They may be used with other stream types as well.

8. RELATED WORK

[5] and [13] describe DMIF-based streaming solutions for MPEG-4. Both implementations carry SL-packetized streams and use TCP for session control. [5] maps SL packets onto UDP. The second project uses RTP/UDP.

Design studies for solutions without DMIF have been published by Liu et al. [14] and Wu et al. [15]. Both concepts use SL packets and RTP/UDP. FlexMux is optional for the first proposal, but mandatory for the second. [14] supports MultiSL too. The second concept suggests source rate control via RTCP-based receiver feedback.

Further open source projects exist. These projects do not consider transparent scaling.

9. CONCLUSIONS

We have introduced methods to map MPEG-4 elementary streams onto RTP packets. These methods differ mainly in the amount of control information that may be associated with ES data and their ability to multiplex several ESs onto one RTP packet stream. We have implemented all three methods and compared implementation effort and runtime needed to execute packetization and depacketization code. Furthermore, we have evaluated mechanisms to couple ES data with metadata that is necessary to steer media adaptation inside the network. Although the needs of powerful network nodes can be satisfied by RTP together with FlexMux packetization, better support for router nodes is required. Hence, we will investigate if it is appropriate to replace these protocols by a new solution that allows to seamlessly integrate router support and stream multiplexing while preserving the useful properties of RTP.

10. REFERENCES

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications (RFC 1889)," Jan. 1996.

[2] K. Nahrstedt, "Quality of Service in Networked Multimedia Systems," in *Handbook of Internet and Multimedia Systems and Applications*, B. Furht, Ed. 1999, pp. 217–252, CRC Press.

[3] R. Koenen, "Overview of the MPEG-4 Standard. ISO/IEC JTC1/SC29/WG11 N4030," March 2001.

[4] ISO/IEC JTC1/SC29/WG11, "ISO/IEC 14496-6:2000(E) Information technology – Coding of audio-visual objects – Part 6: Delivery Multimedia Integration Framework (DMIF)," October 2000.

[5] Y. Pourmohammadi, K.A. Haghighi, A. Mohamed, and H. M. Alnuweiri, "Streaming MPEG-4 over IP and Broadcast Networks: DMIF Based Architectures," in *Proceedings of the 11th International Packet Video Workshop, 30 April - 1 May 2001*, Kyungju, Korea, May 2001.

[6] Y. Kikuchi, T. Nomra, and S. Fukungaga, "RTP Payload Format for MPEG-4 Audio/Visual Streams (RFC 3016)," Nov. 2000.

[7] A. Basso, M.R. Civanlar, P. Gentric, C. Herpel, Z. Lifshitz, Y. Lim, C. Perkins, and J. van der Meer, "RTP Payload Format for MPEG-4 Streams. Internet Draft (draft-ietf-avt-mpeg4-multiSL-04.txt)," Feb. 2002.

[8] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP) (RFC 2326)," Apr. 1998.

[9] M. Handley and V. Jacobson, "SDP: Session Description Protocol (RFC 2327)," Apr. 1998.

[10] J. van der Meer, D. Curet, E. Gouleau, S. Relier, C. Roux, P. Clement, and G. Cherry, "RTP Payload Format for MPEG-4 FlexMultiplexed Streams. Internet Draft (draft-curet-avt-rtp-mpeg4-flexmux-02.txt)," Nov. 2001.

[11] D. Rubenstein, J. Lennox, J. Rosenberg, and H. Schulzrinne, "Bell Labs/Columbia/UMass RTP Library Internal Function Descriptions," Tech. Rep. UM-CS-1999-076, Nov. 1999.

[12] S. Devillers, M. Amielh, and T. Planterose, "Bitstream Syntax Description Language (BSDL). Response to the Call for Proposals on MPEG-21 DIA. ISO/IEC JTC1/SC29/WG11 MPEG/M8273," May 2002.

[13] A. Basso, S. Varakliotis, and R. Castagno, "Transport of MPEG-4 over IP/RTP," in *Proceedings of the 10th International Packet Video Workshop, 1 - 2 May 2000*, Cagliari, Italy, May 2000.

[14] H. Liu, X. Wei, and M.E. Zarki, "A Transport Infrastructure Supporting Real Time Interactive MPEG-4 Client-Server Applications over IP Networks," in *IWDC 2001, LNCS 2170*. 2001, pp. 401–412, Springer-Verlag.

[15] D. Wu, Y.T. Hou, W. Zhu, H.-J. Lee, T. Chiang Y.-Q. Zhang, and H.J. Chao, "On End-to-End Architecture for Transporting MPEG-4 Video over the Internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 6, pp. 923–941, Sept. 2000.