Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Adaptive progressive download based on the MPEG-4 file format

FÄRBER Nikolaus, DÖHLA Stefan, ISSING Jochen[‡]
(*Multimedia Transport, Dept. Audio, Fraunhofer Institute for Integrated Circuits IIS, Erlangen 91058, Germany*)
E-mail: {fae; dla; iss}@iis.fraunhofer.de

Received Dec. 9, 2005; revision accepted Feb. 24, 2006

Abstract: In this paper we describe how progressive download and adaptive streaming can be combined into a simple and efficient streaming framework. Based on the MPEG-4 file format (MP4) we use HTTP for transport and argue that these two components are sufficient for specifying an open streaming architecture. The client selects appropriate chunks from the MP4 file to be transferred based on (1) the header information (i.e. the “moov” box) in the first part of the file and (2) his observation of network throughput. The framework is completely client driven which allows for better server scalability and reduces signalling overhead. We discuss architecture and implementation issues such as complexity, interoperability and scalability and compare to 3GPP PSS Rel-6 adaptive streaming when appropriate. Measurements from a simple MP4/HTTP streaming client are presented showing that appropriate chunks are selected such that increased reliability is achieved.

Key words: HTTP-streaming, Adaptive progressive download, MPEG-4 file format (MP4), Moov box
doi:10.1631/jzus.2006.AS0106 **Document code:** A **CLC number:** TN919.8

INTRODUCTION

In recent years, significant progress has been made in the area of adaptive streaming, i.e., the ability of a streaming system to adapt to varying channel conditions. On the one hand, the challenge of transmitting a continuous stream of data under a real-time constraint over a network with variable throughput, delay and loss has attracted considerable research efforts (Girod *et al*., 2002). On the other hand, the commercial need has led to a quick adoption of adaptive techniques in commercial streaming solutions, see e.g. the SureStream technology by RealNetworks (2004).

However, the ultimate success of IP-based streaming depends on the existence of open streaming standards, such as produced by the Internet Streaming Media Alliance (ISMA) (http://www.isma.tv) or the 3rd Generation Partnership Project (3GPP) (http://www.3gpp.org) for mobile applications. The key goal of open standards is to achieve interoperability between various products from different vendors such that an open market without monopolistic lock-in is assured. Therefore, it is most important that adaptive streaming techniques disseminate from research into open standards in order to be of practical relevance. The lack of such functionality in standard based systems has been addressed most recently with the adoption of RTP retransmission (Ott *et al*., 2004) and adaptive bit-rate adaptation in Release 6 (Rel-6) of the 3GPP Packet Switched Streaming (PSS) specification (3GPP PSS Rel-6). However, the specified solution has practical and conceptual limitations, especially when used outside the mobile domain where it is designed for. In particular for non real-time applications, where pre-encoded files are streamed to the client, we believe that the used architecture is far more complex than necessary.

In this paper we therefore describe an alternative based on the MPEG-4 file format (MP4) (ISO/IEC 14496-14, 2003) and HTTP (Fielding *et al*., 1999) and argue that these two components are sufficient for achieving an open but efficient streaming framework. In Section 2 we describe related work focus on the 3GPP PSS Rel-6 specification and RD-optimal packet scheduling. Then, in Section 3 we describe the adap-

[‡] Corresponding author

tive MP4-streaming framework and discuss several architecture and implementation aspects. Finally, experimental results from a simple MP4/HTTP streaming client are presented in Section 4 showing that increased reliability can be achieved using the described framework.

RELATED WORK

Even though there is a wide range of related work in the area of adaptive streaming, we will focus on two items with particular relevance to our work. One is the 3GPP PSS Rel-6 specification which is used as an example of an RTP based adaptive streaming standard. The second is RD-optimal packet scheduling which provides a theoretical framework for our discussion.

RD-optimized packet scheduling

Recent work of (Chou and Miao, 2001; Chou and Sehgal, 2000) provides a flexible framework to allow rate-distortion (R-D) optimized scheduling of packetized media (http://www.imtc.org). For a layered media stream, providing different quality levels, the framework selects the optimal packet to transmit/retransmit considering channel conditions. This is done by minimizing the Lagrangian cost function of rate and distortion and allocating time and channel resources. The framework shows that two basic components are required as input to the algorithm in order to derive optimal scheduling policies. First, the R-D structure of the media stream, i.e., the gain in quality when increasing the media rate in each temporal segment and the dependencies of these. Second, the channel properties, i.e., the probability that a packet is received after *N* transmission opportunities. While the former is available at the server, the latter has to be measured at the client. Hence, either of both has to be signaled to the other side resulting in client-driven or server-driven approaches. Though both approaches result in optimal solutions, a client-driven approach reduces the complexity at the server and provides better scalability. This is of importance since the algorithm has considerable complexity even after simplifications are applied (Chou and Miao, 2001).

These findings are reflected in our work in that (1) we favour a client-driven approach and (2) use an available open standard to make the structure of the media available to the client, i.e., the "moov" box of the MP4 file.

3GPP PSS Rel-6

Release 6 of the PSS specification (http://www.3gpp.org) was approved by 3GPP in late 2004. It is based on the UDP/RTP/RTCP transport stack (Schulzrinne *et al*., 2003) along with SDP (Handley and Jacobson, 1998) and RTSP (Schulzrinne *et al*., 1998) for signaling and control. For each media format a specific payload format is required, describing how the bit stream is put into packets. For example, the AVC payload format described in RFC 3984 (Wenger *et al*., 2005) is used to build RTP packets from the H.264/AVC (ITU-T Recommendation H.264, 2003) bitstream.

Adaptive streaming functionality is added in Rel-6 by two optional features. First, RTP retransmission (Ott *et al*., 2004) is used to recover from packet loss. In addition, 3GPP has developed a new bit-rate adaptation mechanism in its specification. As part of this mechanism the client signals the total buffer size to the server during setup. Then, during the session the client reports free buffer space and the Oldest Buffered Sequence Number (OBSN) periodically to the server as part of RTCP receiver reports (RR). Together with the Highest Received Sequence Number (HRSN), which is always included in RTCP RRs, the server can obtain detailed knowledge about the buffer state at the client and perform bit-stream switching or thinning to counter variations in network throughput.

The resulting streaming architecture has a conceptual and practical problem. The conceptual problem is that the system is server-driven, i.e., the client is intentionally kept "stupid" while all necessary intelligence is moved to the server. If the server wants to exploit all features to obtain optimal performance, this is actually a quite challenging task. Besides keeping track of the size and content of packets which are not yet acknowledged, the server has to analyze the file structure and reconstruct the state of the client buffer to make RD-optimal switching/thinning decisions. In addition, constraints on the bit stream variations have to be met in order to stay compliant with negotiated profiles. Hence, the architecture does not scale well when several hundreds of clients need to be served.

A practical problem results from the multitude of specifications that need to be implemented simultaneously if interoperability between different vendors is to be achieved. The PSS specification contains more than 20 references directly related to the transport layer (e.g. not including codecs, etc.). Besides the fact that this simply becomes very difficult to implement, which of course can be solved by major players by investing many man months, true interoperability is at risk because any specification leaves room for interpretation. Though interoperability tests at IMTC (ITU-T Recommendation H.264, 2003) and ISMA (http://www.isma.tv) reduce this risk, true N-to-N interoperability requires an immense effort for such a complex system and faces practical limitations. For example, it is still to be seen how bit rate adaptation, RTP retransmission, and AVC interleaving is combined with hint-tracks in an interoperable way.

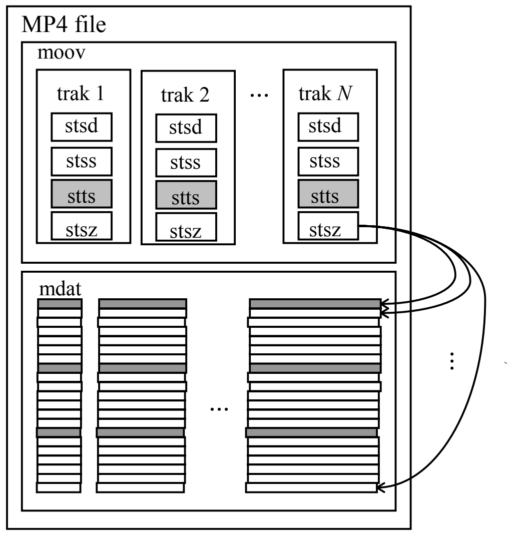
ADAPTIVE MP4-STREAMING

The RTP-based architecture as exploited in the 3GPP PSS specification is very well optimized for the mobile domain and real-time applications. However, when pre-encoded files are transmitted on-demand over general data networks many requirements are void and the resulting architecture is more complex than necessary.

For example, when sufficient time and bandwidth is available for retransmitting lost packets, the delay and loss optimized RTP protocol is "over-designed". There is no need for time stamps and jitter calculation if the dejitter buffer holds several seconds worth of media. Furthermore, sophisticated packetization schemes ranging from video packets to interleaving are not required if retransmission provides a reliable link.

The key to resolve many of the mentioned concerns is a client-driven approach which allows avoiding many signaling and control issues. Furthermore, simplicity is the key to interoperability. Hence we propose an alternative streaming architecture that is based on (only) two open and widely adopted standards, namely the MPEG-4 file format (MP4) (ISO/IEC 14496-14, 2003) and HTTP (Fielding *et al.*, 1999). The basic idea is related to progressive download of an MP4 file using HTTP. However, the file is not loaded linearly from the beginning to the end. Instead, the structure of the MP4 file is used to dynamically select required parts from the file given channel characteristic and/or client capabilities and/or required play mode (play, fast-forward, stop, etc).

In the initial HTTP transfer the client obtains knowledge about the structure of the media based on the MP4 header (i.e. the "moov" box). For each track or media stream in the file, the moov box contains all relevant metadata to decode, play, seek, etc. the media. First of all, this includes the sample description (stsd) which describes which codec is used and how it is configured. Then, the size of each sample (frame) is given in the sample size table (stsz), which can be used to find the location of the sample in the file. Furthermore, the time-to-sample table (stts) provides information about when each sample is played. Hence, the mapping between time and byte-offset is known to the client. Finally, the sync-sample table (stss) indicates which samples can be used for random access and therefore tells the client where switching between tracks is possible. The basic file structure of an MP4 file is illustrated in Fig.1. Note that many more boxes and features are supported in the MPEG-4 file format.



moov: contains all meta data; trak: one track or stream; stsd: sample description (which codec is used in which conf?); stss: synch ample (which samples allow random access?); stts: time to sample (when is each sample displayed?); stsz: sample size (where is sample in file?); mdat: actual media data (samples)

Fig.1 Basic structure of an MPEG-4 file (MP4)

With the knowledge contained in the moov box and based on the continuous observation of the network throughput, the client requests for the appropriate parts of the file that should be transferred next by specifying the byte range in HTTP requests. For this decision he may use an RD-optimized packet scheduling algorithm as mentioned above, however, the decision process within the client does not need to be specified and is open to the implementer.

Data is received from the server through HTTP-GET requests with the support of byte ranges, which is the only interaction between client and server. In order to reduce the overhead of the HTTP header, the MP4 file is structured in relatively large chunks covering 3~10 s of media. This simplistic approach has several interesting consequences that are briefly mentioned below:

(1) Server complexity: Since the server is a simple Web server implementing HTTP 1.1 with the support of ranges, it can be implemented with significantly reduced hardware costs based on existing hardware and software solutions. The fact that individual HTTP-GET transactions are completely stateless also leads to the implementation of server farms.

(2) Content distribution: Existing HTTP proxies can be used as a basis to build very effective Content Distribution Networks (CDN). Hence, an effective alternative to multicasting can be offered based on well established Internet components.

(3) Reduced signaling and control: Since the framework is completely client driven, cumbersome signaling protocols during setup and streaming are avoided. The need for translating setup information from the MP4 file into SDP is avoided since the client reads the file directly. Instead of using an RTSP-pause request, the client simply pauses sending HTTP-GET requests. Even a fast-forward can be implemented by the client by selecting a low resolution track that can be received and decoded faster than real time—without the need of an extra control protocol.

(4) TCP-friendliness and firewall traversal: Since HTTP is based on TCP, these practical problems are resolved implicitly.

(5) Channel measurement: In adaptive RTP streaming the problem of channel measurement often arises. In particular the problem of estimating the available bit rate as required for switching to higher layers requires more than basic RTCP RR measurements. During an HTTP-GET request, however, the channel is automatically probed for the available bit rate using TCP's rate control mechanism. Interestingly, server bottlenecks are treated in the same way as link bottlenecks which is very reasonable.

(6) Media unawareness: No special RTP payload format has to be designed for each new adopted media format.

As mentioned above, TCP-streaming has inherent limitations compared to RTP-streaming, especially when low delay and most effective usage of the available bit rate is of most importance. Wang *et al.*(2004) conclude that "TCP generally provides good streaming performance when the available TCP throughput is roughly twice the media bit rate". However, this finding is derived from the assumption of a constant media bit rate and packet rate. In our approach, the media bit rate is adapted to the available bit rate, i.e., by selecting the appropriate track, the client can assure that the available TCP throughput is twice as big. Our experiments (see below) revealed that even a lower safety margin of 80% is sufficient for continuous media playout.

EXPERIMENTAL RESULTS

In the following we present experimental results from a simple MP4/HTTP client implementation. An example video sequence of 42 s duration is encoded with H.264/AVC at seven quality levels and stored in an MP4 file in separate tracks using alternate groups. We do not use any scalable encoding techniques but simply store redundant encodings at different bit rates using the same spatial resolution. Each track consists of 10 chunks (4 s each) which are started by IDR-Frames to allow switching between tracks. Fig.2 illustrates the bit rate of each chunk showing the VBR nature of encoding. The MP4 file is served from a standard Apache Web server.

The client implements the following simple track selection algorithm: During the previous transfer of a chunk it estimated the available bit rate by dividing the total size of the received chunk by the measured time. This value is reduced by a safety factor (0.8) and the next biggest chunk which allows maintaining of a certain target time in the client buffer is selected for transmission. Note that, due to the VBR

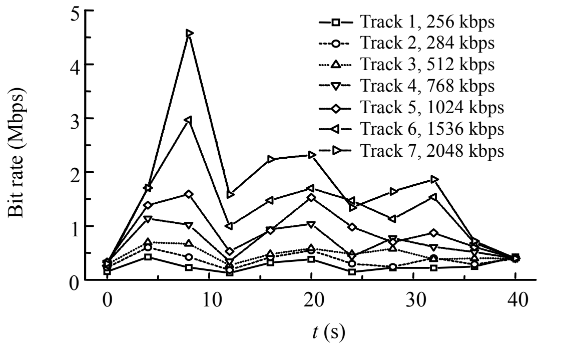


Fig.2 Bit rate of tracks 1~7 for example AVC video sequence used in experiments. Rates in legend are average rate as signaled in moov box

characteristic of the file, this means a content adaptive switching between tracks when the link rate is approximately constant.

Fig.3 shows the transmission of chunks. The width of each rectangle corresponds to the duration of the transmission while the height is proportional to the measured rate during the transfer. The narrow rectangles are from audio tracks which are also transmitted but are ignored here. The maximum link bit rate for this experiment is 1000 kbps. As can be seen, the stream is transferred in bursts during which the transmission rate is increased as much as allowed by the link rate. This allows us to probe the channel repeatedly for the available bit rate.

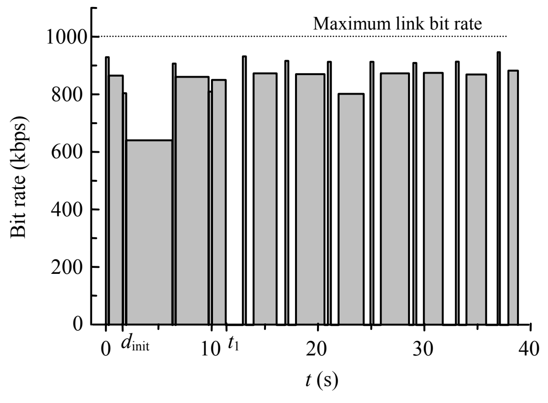


Fig.3 Transmission of audio and video tracks in a burst mode facilitating throughput measurement

This experiment was conducted for various link bit rates of 200 to 2000 kbps, i.e. the link bit rate was artificially limited to 200, 300, 400, … kbps. The resulting distribution of selected tracks for each link rate is illustrated in Fig.4 showing that tracks with higher numbers, i.e. higher quality, are selected more frequently as the link bit rate increases.

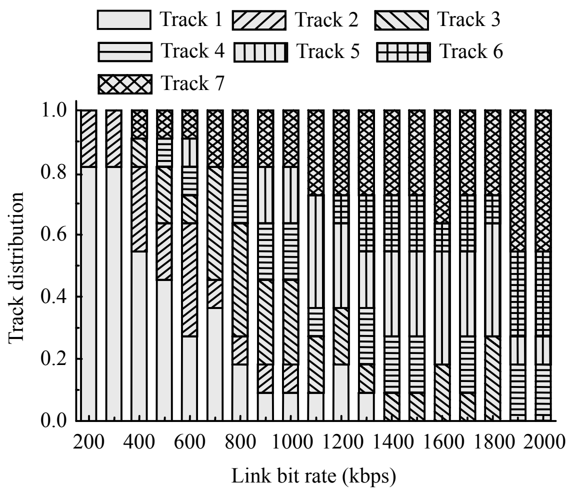


Fig.4 Distribution of video tracks for different link bit rates. Tracks with higher numbers/quality are selected with increasing link bit rate

Finally, Fig.5 shows the probability of buffer under-run for different link bit rates. The figure compares a progressive download of a single track with the proposed adaptive download technique. Obviously, the single track approach fails as the link bit rate drops below the average media bit rate. Only adaptive track switching avoids buffer under-runs over the complete range of link bit rates (at least as long as a small enough media bit rate is available in the file).

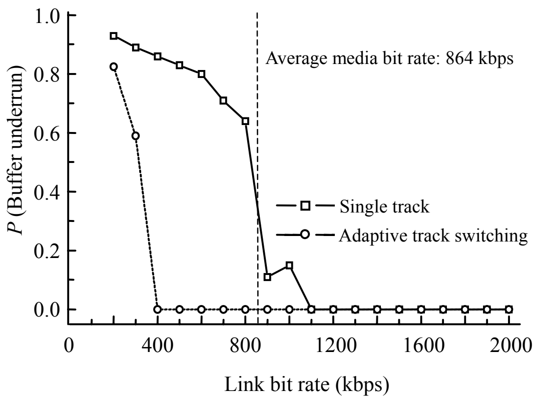


Fig.5 Probability of buffer under-run for different link bit rates. Adaptive track switching assures high reliability over complete range of bit rates

CONCLUSION

For streaming applications where low delay is not of most importance, adaptive streaming techniques based on RTP tend to be more complex than required. On the one hand the complexity of specifications poses a risk to interoperability. On the other hand, the computational requirements on the server are very high and hinder cost effective solutions. In contrast, a simple client-driven approach based on the MP4 file format and HTTP, has conceptual and practical advantages and is easily implemented as demonstrated by initial experiments.

References

Chou, P.A., Sehgal, A., 2000. Rate-Distortion Optimized Receiver-Driven Streaming over Best-Effort Networks. Packet Video Workshop. Pittsburg, PA.

Chou, P.A., Miao, Z., 2001. Rate-Distortion Optimized Streaming of Packetized Media. Microsoft Research Technical Report MSR-TR-2001-35.

Fielding, R., Gettys, J., Mogul, J., *et al*., 1999. IETF RFC 2616: Hypertext Transfer Protocol-HTTP/1.1.

Girod, B., Kalman, M., Liang, Y., Zhang, R., 2002. Advances in Channel-adaptive Video Streaming. Proc. IEEE International Conference on Image Processing ICIP. Rochester, NY, **1**:9-12.

Handley, M., Jacobson, V., 1998. IETF RFC 2327: SDP: Session Description Protocol.

ISO/IEC 14496-10, 2003. Information technology—Coding of Audio-Visual Objects—Part 10: Advanced Video Coding.

ISO/IEC 14496-14, 2003. Information Technology—Coding of Audio-Visual Objects—Part 14: MP4 file format.

ITU-T Recommendation H.264, 2003. Advanced Video Coding for Generic Audiovisual Services.

Ott, J., Wenger, S., Sato, N., *et al*., 2004. IETF Internet Draft: Extended RTP Profile for RTCP-based Feedback (RTP/AVPF). Http://www.ietf.org/internet-drafts/draft-ietf-avt-rtcp-feedback-11.txt.

RealNetworks, 2004. RealNetworks Production Guide. p.49-51.

Schulzrinne, H., Rao, A., Lanphier, R., 1998. IETF RFC 2326: Real Time Streaming Protocol (RTSP).

Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., 2003. IETF RFC 3550: RTP: A Transport Protocol for Real-Time Applications.

Wang, B., Kurose, J., Shenoy, P., Towsley, D., 2004. Multimedia Streaming via TCP: An Analytic Performance Study. Proceedings of ACM Multimedia (Multimedia 2004). New York.

Wenger, S., Hannuksela, M.M., Stockhammer, T., *et al*., 2005. IETF RFC 3984: RTP Payload Format for H.264 Video.