A PROTOCOL FOR ADAPTATION-AWARE MULTIMEDIA STREAMING

Matthias Ohlenroth and Hermann Hellwagner

Department of Information Technology
University Klagenfurt
Universitätsstraße 65-67
A-9020 Klagenfurt, Austria
{Matthias.Ohlenroth, Hermann.Hellwagner}@itec.uni-klu.ac.at

ABSTRACT

Multimedia streaming is becoming ever more popular. However, the Internet does not support streaming with its high bandwidth and low latency requirements very well. The problem is that QoS guarantees cannot be given. Hence, communication partners have to deal with rapidly changing connection parameters. This requires sophisticated streaming concepts that can handle these varying conditions using adaptation techniques. Adaptation methods can be dropping layers, dropping access units or transcoding the contents. But this places specific requirements on the underlying protocol. This paper identifies and discusses these requirements and analyzes how existing protocols can meet them. Unfortunately, none of the known protocols can meet all requirements. Hence, we propose a new adaptation-aware multimedia streaming protocol that can operate as required in the given Internet environment. Furthermore, we show how this protocol can be used to carry MPEG-4 audio-visual contents.

1. INTRODUCTION

Multimedia streaming allows a client to start playing as soon as a sufficient portion of the video has been received. But this depends heavily on the network conditions during play-back. The streaming mode requires certain quality of service (QoS) guarantees from the networks traversed. 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 can 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. Three 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. This method is called *non-transparent* video scaling as it involves both client and server [2].
- The second method uses layered encoding together with multiple multicast groups. A receiver joins and leaves multicast groups depending on the congestion it observes [3]. Hence, adaptation is controlled by the receiver.
- The third method shifts the video adaptation task into the network [4] 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].

Our research attempts to use the scalability features provided by MPEG-4-encoded multimedia presentations together with an appropriate streaming protocol to enable the network to perform transparent video scaling, especially if short-term congestion events occur.

The paper is organized as follows. First, we present requirements that must be satisfied by protocols capable of supporting transparent video scaling. Based on this, we evaluate how state-of-the-art protocols satisfy these requirements and show that several requirements cannot be fulfilled properly. Consequently, we introduce a new protocol and describe its transparent video scaling features. Next, we show how this protocol can be used to carry MPEG-4 contents and compare its performance to existing solutions. The last section provides concluding remarks.

2. REQUIREMENTS ON ADAPTATION-AWARE MULTIMEDIA STREAMING PROTOCOLS

Transparent video scaling relies on the capabilities of scaling nodes that can be cascaded. These scaling nodes are distributed in the network and act as routing nodes or as proxies with scaling functionality for their outgoing network links. Scaling nodes can be divided into two groups:

- **Scaling routers.** The function of scaling routers extends the function of ordinary routers. They are able to perform scaling actions on an active multimedia transmission that requires low computing power and limited or no state memory. Such scaling actions may be dropping packets or dropping packet groups.
- **Scaling gateways and proxies.** These nodes may perform scaling router tasks, but they have much more processing power and memory that enables them to perform more complex scaling tasks like transcoding.

Scaling nodes are assumed to become part of the IP networks that route multimedia streams between server and clients. Clients like mobile phones or PCs have different properties and hence, they have different bandwidth requirements. Scaling nodes base their actions on scaling hints provided by the server and on current network QoS conditions.

Based on this model the following requirements arise:

- **Unicast and multicast support.** Multicast helps saving bandwidth as compared to multiple unicast connections.
- **Transfer of configuration information in-band.** The protocol to be used should be self-contained. All information needed to perform scaling actions should be available by decoding protocol header information and/or special configuration packets. This allows routers to become part of the processing chain because they are not able to manage separate configuration "channels". Periodic retransmission of this information allows clients to join ongoing streaming sessions.
- **Multiplexing of streams.** Several streams belonging together like base layer and enhancement layer video should be multiplexed into one session. This simplifies the task of scaling nodes because it is not necessary to maintain special state that relates different sessions together. Furthermore, it must be possible to mark streams as not to be modified.
- **Delivery of scaling guidelines.** The protocol should support the transmission of information that steers the scaling decision if several actions are available.
- **Scaling actions must preserve standards conformance of streams.** Scaling nodes should only perform actions that do not change streams in a way that makes them undecodable to standards-conforming clients.
- **Support of scaling nodes with different processing capacities.** The protocol should transmit information that allows scaling routers, gateways and proxies to act according to their abilities.
- **Scaling guided by current network QoS conditions.** QoS information may be directly available to router software from local measurements or may be computed by the protocol software and communicated upstream.

3. ANALYSIS OF EXISTING MPEG-4 STREAMING CONCEPTS

Streaming of MPEG-4 encoded audio-visual content over IP networks could be realized using the RTP protocol together with packetization modes. Several packetization modes are under development or have been finalized [5, 6, 7, 8].

RTP supports unicast and multicast, but it does not support broadcast because an upstream connection is required by RTCP. QoS information can be computed by client nodes and server nodes using the RTCP sender reports and receiver reports, respectively. RTCP SDES or APP packets can be used to carry configuration information in-band. Unfortunately, scaling nodes must establish relations between RTP and corresponding RTCP connections to use this information. The packetization modes defined by [5], [6] and [7] do not support the multiplexing of streams. Stream multiplexing and hence the delivery of scaling guidelines using a separate stream is supported by [8]. However, extracting scaling guidelines out of such RTP streams requires knowledge about the mapping of media streams onto RTP PDUs and knowledge about the payload-specific header. Hence, scaling nodes have to rely on out-of-band configuration information. To support scaling nodes with limited CPU power, the Differentiated Services model (DiffServ) [9] can be used. However, the assured service per-hop behavior [10] does not allow precise steering of the packet dropping mechanism, only dropping probabilities can be specified.

The concept of thin streams [11] allows to split one media stream consisting of media access units that contribute differently to the perceived media quality at the client onto a set of multicast sessions. A client connects to the lowest layer first. Connections to the higher layers are established and maintained as long as the loss rate is below a threshold. Unfortunately, this mechanism provides no information to scaling nodes in the network to help them select appropriate packets for dropping.

Thin streams are a suitable basis for network-based adaptation as long as routers receive information about how to intelligently drop packets in case of congestion [4].

The stream control transmission protocol [12] provides a mechanism to multiplex several streams into one session. Data reliability is a key feature of this protocol, but is not helpful for video streaming. However, protocol-level congestion control may render application-level packet scheduling useless.

Although several requirements could be fulfilled at least partly, support for transparent scaling is difficult to achieve with the protocols described. It is even more difficult to deliver scaling guidelines to scaling routers because there is no way to integrate such information into already defined header structures in a way that would enable scaling routers to extract this information efficiently. Hence, we propose a new streaming protocol that explicitly carries scaling information.

4. THE ADAPTATION-AWARE MULTIMEDIA STREAMING PROTOCOL

4.1. General Concept

Like RTP, the adaptation-aware multimedia streaming protocol (AMSP) is based on the principle of application layer framing [13]. It establishes communication sessions between two or more endpoints. One session connects one sending node (called server) and one or more client nodes that receive data. A session typically manages multiple data flows between server and clients. These data flows are organized as channels and transmit control information, media objects, and corresponding metadata. A media object consists of a single media stream or a group of related media streams that can be mapped onto a set of prioritized media channels. The organization of the protocol facilitates media scaling to be performed by network nodes, e.g. scaling nodes.

4.2. Channel Concept

To separate layered streams, AMSP provides the concept of *prioritized channels*. Each stream out of a set of related multimedia content streams is mapped to one (simple mode) or more (fine granular mode) media channels (MCs). The scaling model of the AMSP protocol assumes that lower priority channels should be dropped first. Hence, more important media layers should be mapped to higher priority channels, i.e., channels with lower identification numbers.

Metadata channels (MDCs) are defined to transmit metadata like MPEG-7 descriptions. Scaling control channels (SCCs) are used to transmit scaling hints needed for complex scaling actions like transcoding. Auxiliary channels (AUXs) allow servers the inclusion of further streams (e.g. audio and MPEG-4 Systems streams) into a session that transmits scalable video contents. This simplifies client design because the number of AMSP sessions that must be maintained concurrently is reduced.

The control channel (CC) is a predefined channel that is used to send control and configuration information like the usage of other channels in-band. This simplifies the implementation of scaling routers because it is not necessary to manage separate control connections and to handle the relationships between multiple connections.

The *simple mode* allows to establish a one-to-one relationship between streams and AMSP channels. In case a media object consists of only one or a few elementary streams (ESs), scaling decisions suffer from a coarse granularity due to the limited number of streams. To overcome this, we provide a mechanism to map such objects onto a set of channels in a regular way (see figure 1). A mapping module implemented by the application has to divide a set of incoming ESs belonging to the media object into access units (AUs) or fragments thereof that can be mapped onto AMSP packets. These packets are forwarded onto a bundle of channels such that higher-priority packets (i.e. packets that should be discarded with lower preference) will be assigned to lower-numbered channels.

Figure 2 illustrates this using an MPEG-4 Visual ES that contains I-, P- and B-Frames. In this example, it is assumed that scaling in network nodes consists of dropping frames. I-Frames will be referenced by all other frame types. Hence, these frames should not be dropped. P-Frames will be referenced by B-Frames. Hence, they should be mapped to the next MC. B-Frames will be distributed over the remaining MCs such that holes in the packet flow due to frame dropping occur at nearly regular intervals. In this case, two further MCs are appropriate.

In *fine granular mode*, each coded AU of an MPEG-4 Visual FGS ES could be split such that groups of bitplanes are mapped onto separate AMSP packets that are inserted into MCs corresponding to the importance of these bitplane levels. This mapping mechanism enables even scaling routers to perform scaling actions with a much finer granularity than the simple mode would allow in this case. Furthermore, because scaling routers act at the level of channels they do not need further information about the encoding or type of the media data.

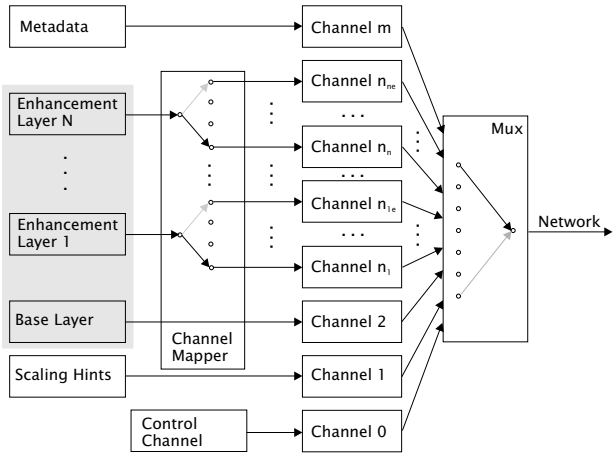

Fig. 1. AMSP channel concept

To enable clients to join existing multicast or broadcast sessions that are running, channel setup messages will be sent periodically. Hence there is no need to establish session control via RTSP or similar protocols for such situations.

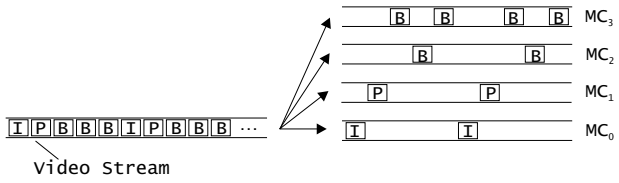

Fig. 2. Mapping video AUs onto a set of channels

AMSP specifies only the encoding of the common parts of the CC. The encoding of all other channels must be defined by channel specifications. A channel specification defines the mapping of encoded AUs onto messages carried by a specific channel. This may depend on the encoding of the media data to be transmitted.

A channel specification that allows the mapping of synchronization layer (SL) [14] encapsulated MPEG-4 AUs onto AMSP packets has been defined.

5. SCALING AMSP SESSIONS

We propose two methods to scale media streams inside the network. The first method is based on the channel concept. Scaling routers simply drop channels if the bandwidth of the outgoing link is lower than the bandwidth that the full AMSP session would require. If a set of consecutive B-Frames is mapped onto a set of channels, channel dropping would result in the dropping of some of the B-Frames. The second method is based on media transcoding. Scaling proxies receive encoded scaling hints. Based on this information, these nodes may perform more complex media transformations. The first mechanism is independent of media encoding standards. The second mechanism depends on the media encoding used because transcoding does as well.

6. QOS FEEDBACK AND RETRANSMISSION

AMSP allows clients and scaling nodes to send QoS feedback to the server. The server may permit a limited set of clients to send ACK or NACK messages together with retransmission requests. The server answers such requests using dedicated retransmission channels. To achieve a reasonable interleaving of priorities, each ordinary channel is assigned to one retransmission channel that has a priority level directly above the priority of the ordinary channel. Hence, priority is given to important packets and important retransmission packets.

7. PERFORMANCE EVALUATION

An initial implementation of AMSP has been developed. This includes AMSPlib, a C++-based implementation of the protocol, and AMSProuter, a module that fits into the Linux 2.4.x router and traffic control framework.

First, we compare AMSPlib to UDP and RTP-based streaming solutions for MPEG-4 elementary streams. All implementations are our own solutions. Preliminary performance results (see figure 3) show that AMSPlib performs well. However, further optimization of the library is appropriate, because there is a significant performance gap to UDP and RTP especially for small packets.

Please note that RTP together with FlexMux packetization performs poorly because simple mode was used and each SL packet has been mapped onto one RTP packet.

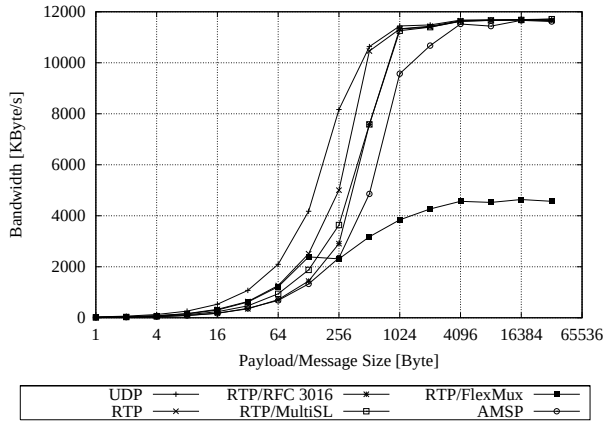

Fig. 3. Performance of AMSPlib compared to UDP and RTP

The second test demonstrates the ability of the AMSProuter to drop packets according to their priority. To achieve this, a 512 kBit/s stream with 30 fps has been sent. The size of the frames has been chosen such that I- and P-Frames including headers fit into 64 kBit/s and that B-Frames with relative priority 3 fit into another 64 kBit/s. Fragmentation was done by AMSPlib. The outgoing link of the router was limited to 512 kBit/s, 256 kBit/s, 128 kBit/s and 64 kBit/s for approximately 10 s each. Figure 4 shows one dot for each AU that has been received. Incomplete AUs are ignored. As expected, AMSProuter honors packet priorities very well.

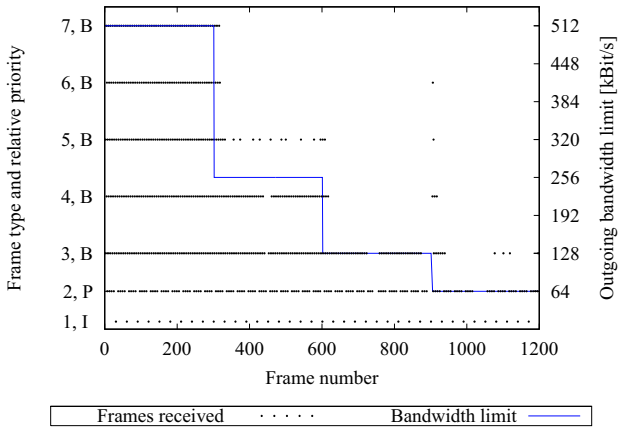

Fig. 4. AMSProuter handling bandwidth limitations

8. CONCLUSIONS AND FURTHER WORK

As shown in this paper, state-of-the-art protocols and techniques for multimedia streaming do not satisfy all requirements of media scaling. Therefore, we have developed and presented a new multimedia streaming protocol (AMSP) with built-in support for transparent video scaling. The results presented show that network-based transparent scaling is feasible within the IP-based Internet. Our future work focuses on QoS feedback and retransmission.

9. 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] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast," in *Proceedings of ACM SIGCOMM 96*, New York, Aug. 1996, vol. 26,4, pp. 117–130, ACM Press.

[4] S. Bhattacharjee, K. Calvert, and E. Zegura, "Network support for multicast video distribution," 1998, Technical Report 98-16, Georgia Institute of Technology, College of Computing.

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

[6] 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.

[7] J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, and P. Gentric, "RTP Payload Format for Transport of MPEG-4 Elementary Streams. Internet Draft (draft-ietf-avt-mpeg4-simple-07.txt)," Feb. 2003.

[8] 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-03.txt)," July 2002.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services (RFC 2475)," Dec. 1998.

[10] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group (RFC 2597)," June 1999.

[11] L. Wu, R. Sharma, and B. Smith, "Thin Streams: An Architecture for Multicasting Layered Video," in *Proceedings 7th Int'l. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, St. Louis, USA, May 1997.

[12] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream Control Transmission Protocol (RFC 2960)," Oct. 2000.

[13] D.D. Clark and D.L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, Philadelphia, Pennsylvania, Sept. 1990, pp. 200–208.

[14] *ISO/IEC 14496-1:2000(E) Information technology – Coding of audio-visual objects – Part 1: Systems. ISO/IEC JTC 1/SC 29/WG 11 N3850*, 19 October 2000.