MODIFICATION PATTERNS FOR INTELLIGENT TEMPORAL VIDEO SCALING IN AN ADAPTIVE ENVIRONMENT

Michael Kropfberger, Klaus Leopold, Hermann Hellwagner

Department of Information Technology
University Klagenfurt, Austria
{mike, klaus, hellwagn}@itec.uni-klu.ac.at

ABSTRACT

This paper discusses adapting (scaling) an MPEG-4 video to comply with a given (or changing) network bandwidth. We differentiate between realtime and non-realtime video adaptation, which is to be performed by routers and (proxy) servers, respectively. Mainly we will look into temporal video scaling, which simply means dropping the least important frames. This can be done on the fly, in a realtime manner, on a router with extended functionality.

To do so, we have to look into the patterns of frames existing in a video. Those patterns (and correspondingly, the video data) can be modified in different ways such that, e.g., varying network bandwidth can be closely matched. Such *modification patterns* and their effectiveness (in terms of bandwidth reduction) must be communicated to the routers in order to enable them to take "intelligent" scaling actions.

We describe the concept of modification patterns and show how they are periodically communicated to, and utilized by, routers for scaling and transmitting a video stream. Furthermore, we introduce an efficient binary representation for them.

1. INTRODUCTION

Streaming video in a best effort network environment has to quickly adapt to changing bandwidths. This adaptation has to happen as close to the clients as possible, so we work on an adaptive video streaming architecture not only including a video server, but also active routers and video caching proxies. The latter two components are to be enabled to operate on video streams and adapt them to the current network situation, the requesting terminals' capabilities, or the users' preferences.

The easiest and fastest way of adaptation is to simply drop the least important frames. This paper will show new ideas on generating more sophisticated adaptation patterns on this basic idea and outline an overall environment how these new adaptation patterns can be transferred onto a real best effort multicast network.

2. VIDEO ADAPTATION METHODS

MPEG-4 [1] supports different profiles [2] specifying how video might be coded and furthermore scaled to different qualities. Concerning adaptability and scalability in the context of video transmission over networks, we have to differentiate between realtime and non-realtime scaling of video streams. Realtime methods can be handled on a router while forwarding the packets containing video data. Non-realtime adaptation methods, e.g., more extensive transcoding, have to be performed at the video server or a proxy server. In this paper, we will focus on realtime methods.

Realtime scalability describes all possibilities for intermediate devices on the way between the video server and the client, to react and adapt to the actual network load or other environmental influences, always without delaying the video playback. Methods for realtime scalability are:

- Temporal Scalability
- Spatial Scalability
- Quality (SNR) Scalability
- Fine Granular Scalability (FGS) [3]

In the following, we will focus on efficient temporal video scaling. However, the principal ideas will be applied to other realtime scalability methods in future work.

3. MODIFICATION PATTERN GENERATION

Concerning MPEG-4 video elementary streams [4], three video frame types are distinguished: I-frames, P-frames, and B-frames. I-frames are independent from any other frames, P-frames are based on predictions from the last reference frame, and B-frames are based on predictions from the previous and the following reference frames. A reference frame might be either an I-frame or a P-frame, so only B-frames are totally unreferenced by any other frame type.

If we look at a video bitstream, we can detect a pattern starting with an I-frame followed by subsequent frame types which might look like `IBPBPB`. This pattern might be repeated over the whole video, or there will be different patterns encoded to immediately react to scene changes. So the first frame after a scene cut should be an I-frame, with any combination of following referencing frames.

Temporal scalability allows us to simply drop frames from a stream. It is the decoder's duty to interpolate over missing frames or at least to keep playing out the latest frame until the next available frame can be presented.

The influence on the video by dropping a certain frame type can be summarized as follows:

- B-frames can be dropped at will since there are no other frames referencing them.

- When dropping a P-frame P_i, all previous B-frames forward-referencing P_i and all following $P_{i'>i}$ and B-frames have to be dropped, too:
  `IPBBPBBPBB(P)BBPBBI` $\rightarrow$ `IPBBPBBP(BBPBBPBB)I`
- Dropping I-frames means losing all following P- and B-frames until the next I-frame, and again all forward referencing B-frames. Since I-frames are used rarely in a frame pattern, dropping I-frames makes nearly no sense.

To add more flexibility for possible adaptations, MPEG-4 also supports a two-layer approach, where the *base layer* stores, e.g., `I-B-P-B-P-B-` and the *enhancement layer* stores `-P-B-B-P-B-B`. When those two layers are interleaved, the received pattern would be `IPBBPBBPPBBB`. Since the enhancement layer is totally independent, we can also drop the whole enhancement layer (probably including P-frames) without interfering with any P-frames stored in the base layer.

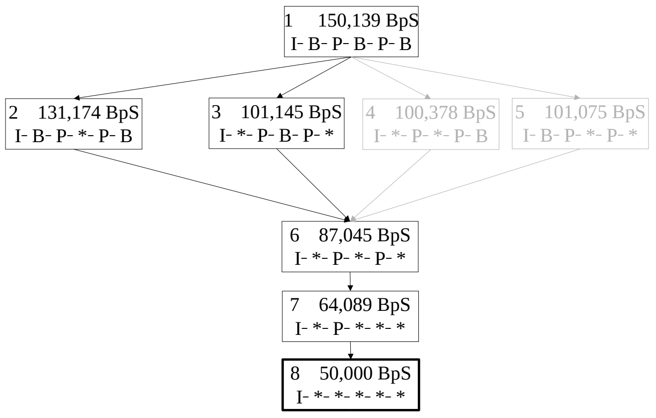

Figure 1: Tree of possible base layer modification patterns

Figure 1 shows a possible (non-exhaustive) modification tree on a base layer pattern where some combinations are grayed out, according to the following heuristics:

- Importance $(I > P > B)$
- Timely balanced distribution:
  - Pattern `I-*-P-*-P-` better than `I-B-P-*-*-`
  - Skipping the whole enhancement layer might even be better than `P-*-*-*-*-*` (not worth the effort)
- Averaged signal-to-noise ratio (SNR) for modified patterns
- Tree size vs. accuracy (grain) of scaling:
  - Tree pruning with respect to the above mentioned heuristics like timely balanced distribution
  - Removal of similarly sized nodes based on threshold values. Possible decision rules are:
    * Preferring higher frame rates
    * Preferring higher-quality patterns; e.g., (`I-*-P-*-P-` is better than `I-B-*-B-*-`)
- and finally, qualitative knowledge gained from MPEG-7 meta-data descriptions [5], for example important scenes, which are represented by a sequence of important frames.

So these grayed out modifications will not be included in the *pool of possible modification patterns*. Only that reduced pool can be utilized by the routers to select adaptations matching detected bandwidth. Since we know the frame sizes for I-, P- and B-Frames occurring in this pattern, we can store accumulated pattern bandwidth requirements for each node. These bandwidth figures are the main selection criteria for routers to choose a specific modification.

The enhancement layer could be modified with the same rules as the base layer tree, except the final node might be an empty one, which means skipping the whole enhancement layer.

Dealing with layer-encoded streams will add further complexity and adaptation restrictions to the above mentioned approach, since the enhancement layer might refer to reference frames in the base layer. Furthermore, pruning has to be done with respect to the achievable "quality" of combinations of base and enhancement layer patterns.

4. REALTIME STREAM ADAPTATION

To be efficient in the routers, we have to break down the already pruned trees into lists or binary trees sorted by the needed bandwidth per pattern (bits per second). These *modification pools* will be transmitted to the downstream routers, so they can choose the best fitting pattern modification. Video frames are sent via RTP [6] using a standardized packet format for MPEG-4 [7].

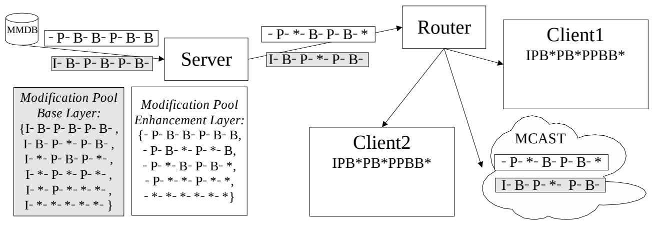

Figure 2: Clients requesting the same video quality

The *Server* reads the base layer and enhancement layer from disk and only sends the requested frames to the next *Router* hop. This next *Router* requests the maximum frame pattern of all connected clients (or downstream routers). If, as is the case in Figure 2, the clients want the same quality, the adapted base and enhancement layers are sent via multicast. To keep network traffic low, we share as much information as possible in the multicast layers.

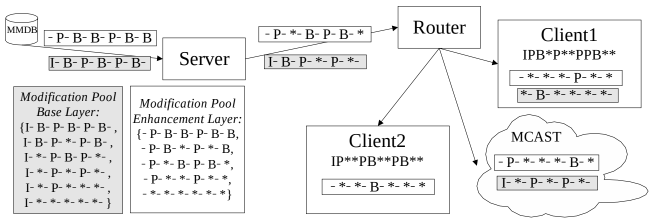

Figure 3: Clients requesting different video quality levels

If clients have individual needs, as exemplified in Figure 3, the *Router* generates shared base and enhancement layers and extracts two new enhancement layers and one base layer adjusted for the two clients. Since MPEG-4 compliant players are only capable of one base layer and one enhancement layer, we have to add an intelligent client-side “network tunnel” that puts together multicasted and unicasted frames into a correct base and enhancement layer format, which is eventually fed to the player.

5. COMMUNICATION PROTOCOL IN AN ADAPTIVE ENVIRONMENT

1. A new *Client* sends initial specification of video name, resolution, environment, and more to the *Router*.
2. *Router* requests for the initial video information like resolution, colordepth, average bandwidth and average bandwidth reached by maximum adaptivity.
3. *Server* sends first the *modification pool* with bandwidth requirements.
4. *Router* checks available bandwidth to *Clients*.
5. *Router* requests necessary maximum frame pattern to serve all connected *Clients*.
6. *Server* sends out the requested frame pattern and the next *modification pool* (see Figure 4).
7. *Router* sends out the shared base and enhancement layer frames via multicast.
8. *Router* generates the specific base and enhancement layers and sends them to the connected *Clients* via unicast.
9. If no new clients arrive, go back to Step 4.

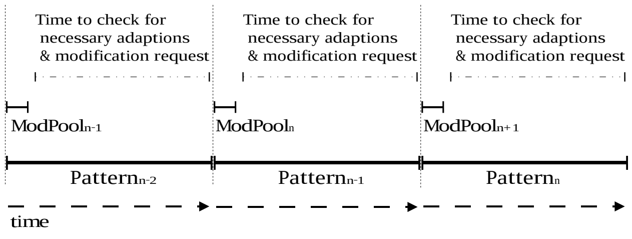
Time to check for necessary adaptions & modification request
Time to check for necessary adaptions & modification request
Time to check for necessary adaptions & modification request
ModPool$_{n-1}$
ModPool$_n$
ModPool$_{n+1}$
Pattern$_{n-2}$
Pattern$_{n-1}$
Pattern$_n$
time

Figure 4: Modification pool MP_n is sent with pattern P_{n-1}, so there is enough time to react to changing bandwidth requirements

6. EFFICIENT BINARY REPRESENTATION OF ADAPTATION DESCRIPTIONS

Presently our high-level model to describe modification patterns is based on XML, like in MPEG-7 [5] and proposed for the MPEG-21 Bitstream Syntax Definition Language [8] [9].

The communication model contains

- *pre-sent video information*, giving first information about the video concerning resolution, min/max bandwidth needs and general modification choices.
- *per-pattern information packets*, including modification patterns from modification tree, according bandwidths and drop lists.
- *the modification choice*, which is the local maximum of required client needs fitting a pattern from the modification pool.

A detailed description with XML examples is given in [10].

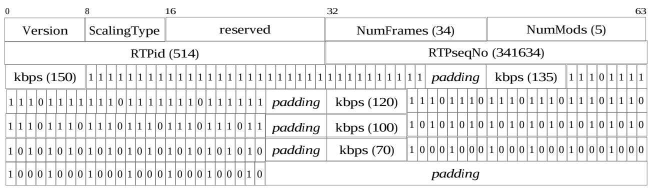

Figure 5: Packet format for per-pattern modification pool

Since we cannot directly handle XML files on a router because of performance issues, we have to devise a highly efficient binary representation of at least the periodically sent per-pattern modification pool. This low-level packet layout (Figure 5) is preferred, because with our approach we keep the packet size at a minimum. Other devices might still receive XML information.

7. IDEAS ON OPTIMIZATION

Even with the efficient packet layout for the regularly sent pattern modification packets, we will add an average network load of approximately 60-100 bytes per pattern. In normal cases we can easily ignore this overhead, but on very low bandwidth networks, we would like to further reduce the amount of data to send or process.

Given the case that

- a video uses the same pattern repeatedly over the whole lifetime (or at least for a couple of iterations),
- the average needed bandwidth per pattern is very similar,
- and the average frame sizes between and within the patterns are similar,

we could merge multiple patterns into a *pattern group*. Each pattern group will hereby represent, e.g., the next fifty patterns and we calculate the pattern modification tree based on this averaged pattern group.

After building a list of very few pattern groups reflecting the whole video, we can send these few pattern modification lists within the initially transmitted information. No further updates are necessary.

Unfortunately our measurements on different videos with multiple qualities, patterns, and MPEG-4 encoders revealed that we cannot rely on the above mentioned assumptions at all.

There are too high variations already on a frame-to-frame basis, which make a pattern group, given a reasonable threshold, too small and patterns which seemed to belong to one pattern group by having the same average bandwidth, would differ from each other after applying modifications, which means dropping frames on certain positions in the pattern (see Figure 6).

8. AVAILABLE SOFTWARE

We wrote a statistic suite capturing frame types and sizes, average deviations with threshold triggers. We can detect patterns over a video, again with average bandwidths and exact frame sizes,

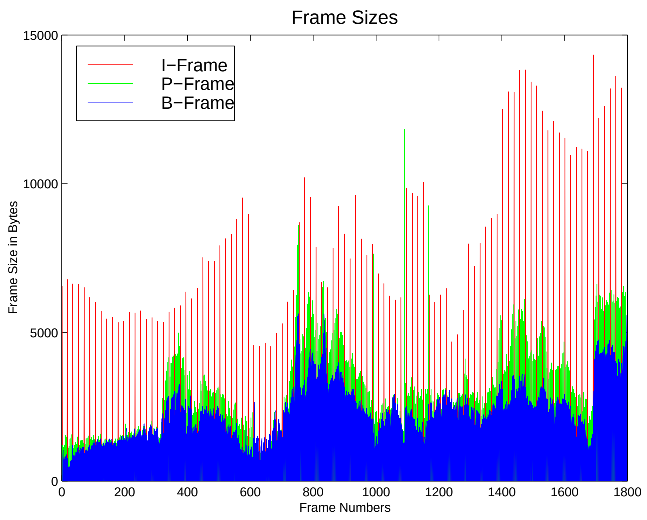

Figure 6: Frame size variations

threshold triggers to indicate deviation to the prior patterns. We aggregate patterns into pattern groups and we generate averaged frame sizes and bandwidth requirements.

We support tools to simulate videos with dropped frames. For control issues we have a tool to merge dropped frames back into the original video.

Finally we have an implementation for transmitting MPEG-4 elementary streams over RTP networks.

For a more detailed description please refer to [10] and [7].

9. RELATED WORK

The problem concerning best effort networks and multicast video distribution is well known. So there is work on scaling in active routers [11] with introduced extra latency per hop. Other ideas discuss splitting of a video stream into multiple “thin” streams [12] [13] without specifying a certain video codec. Splitting an MPEG-4 stream into independent streams would lead to “thick” streams, though. Anyway, thin streams are only sent on demand to multicast subnets.

The importance of bi-directionally predicted frames (B-frames) for scalability and packet loss is discussed in [14].

10. CONCLUSION AND FUTURE WORK

In this paper we outlined different types of scaling an MPEG-4 video to adapt to network bandwith changes. We described the internal representation of an MPEG-4 video bitstream and introduced the idea of pattern generation and modification of frame groups to do realtime adaptation of a video stream on a router. We outlined a communication protocol in an adaptive environment based on XML and described an efficient representation of the XML content so it can be handled by a router.

Presently we are working on intelligent algorithms to generate patterns to massively improve the quality of the video on clients despite dropping frames on a router. Furthermore, we are trying to deploy other methods for video scaling on a router different from temporal scaling.

11. REFERENCES

[1] R. Koenen, “Overview of the MPEG-4 Standard,” *ISO/IEC JTC1/SC29/WG11 N4030*, March 2001. http://mpeg.telecomitalialab.com/.

[2] R. Koenen, “Profiles and Levels in MPEG-4: Approach and Overview,” *Image Communication Journal. Tutorial Issue on the MPEG-4 Standard*, vol. 15, January 2000. http://leonardo.telecomitalialab.com/icjfiles/mpeg-4_si/.

[3] W. Li, “Overview of Fine Granularity Scalability in MPEG-4 Video Standard,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, March 2001.

[4] C. Herpel and A. Eleftheriadis, “MPEG-4 Systems: Elementary Stream Management,” *Image Communication Journal. Tutorial Issue on the MPEG-4 Standard*, vol. 15, January 2000. http://leonardo.telecomitalialab.com/icjfiles/mpeg-4_si/.

[5] J. M. Martinez, “Overview of the MPEG-7 Standard,” *ISO/IEC JTC1/SC29/WG11 N4031*, March 2001.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RFC 1889: RTP: A Transport Protocol for Real-Time Applications,” January 1996.

[7] M. Ohlenroth and H. Hellwagner, “RTP Packetization of MPEG-4 Elementary Streams,” Tech. Rep. TR/ITEC/02/1.01, University Klagenfurt, February 2002.

[8] S. Devillers, “Bitstream Syntax Definition Language (BSDL): An Input to MPEG-21 Content Representation,” *ISO/IEC JTC1/SC29/WG11 M7053*, March 2001.

[9] A. Vetro, “MPEG-21 Requirements on Digital Item Adaptation,” *ISO/IEC JTC1/SC29/WG11 N4515*, December 2001.

[10] M. Kropfberger, K. Leopold, and H. Hellwagner, “Investigations Into Efficient Temporal Video Scaling,” Tech. Rep. TR/ITEC/02/1.02, University Klagenfurt, February 2002.

[11] R. Keller, S. Choi, D. Decasper, M. Dasen, G. Fankhauser, and B. Plattner, “An Active Router Architecture for Multicast Video Distribution,” in *IEEE Infocom 2000*, (Tel Aviv), March 2000.

[12] L. Wu, R. Sharma, and B. Smith, “Thin Streams: An Architecture for Multicasting Layered Video,” in *NOSSDAV’97*, May 1997.

[13] R. Rejaie, M. Hanley, and D. Estrin, “Layered Quality Adaptation for Internet Video Streaming,” *IEEE JSAC. Special Issue on Internet QoS*, Winter 2000.

[14] T. Shanableh and M. Ghanbari, “The Importance of the Bi-Directionally Predicted Pictures in Video Streaming,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, March 2001.