KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 298–306

EVALUATING DYNAMIC CLIENT-DRIVEN ADAPTATION DECISION SUPPORT IN MULTIMEDIA PROXY-CACHES

ADRIAN STERCA, CLAUDIU COBÂRZAN, FLORIAN BOIAN, DARIUS BUFNEA

Abstract. Adaptation of multimedia streams in proxy-caches by usually lowering their quality as a result of a transcoding operation, can yield a lot of benefits in situations when resources like available bandwidth are scarce. Such an operation becomes mandatory when client preferences and/or display capabilities have to be met. We evaluate an alternative to the static specification of terminal capabilities and user preferences inside multimedia proxy-caches, namely the use of *scaling hints*, provided by a protocol still under development, which enables the client to dynamically indicate to proxy-caches or origin servers the appropriate course of action based on network load or user's desires.

1. Introduction

Streamed multimedia data, which continuously gains a bigger percent of the total amount of data transferred over the Internet, tends to stress the existing infrastructure as it demands huge bandwidths (when compared for example with web traffic) and low latencies. Because in most cases, no QoS guaranties can be given, in the rapid fluctuating environment which is the current Internet, availability of multimedia services is certainly a problem. After a successful deployment in the World Wide Web domain, proxy-caches have also been used in the area of multimedia communications (for video on demand services, video broadcasting, etc.). A multimedia cache holds multimedia objects (e.g. movies, audio streams, animations, presentations, etc.) requested by clients in the hope that future requests can be serviced using the already retrieved objects. Such a mechanism can alleviate the availability problem, because multimedia objects are served from the cache through a much better connection (higher bandwidth, smaller delay and jitter), instead of being served from the original streaming server through a fluctuating network line. The gains are not only on the client side but also on the server as an efficient proxy-caching service helps diminish the load imposed on the servers.

2000 *Mathematics Subject Classification.* 68M14, 68M12.
Key words and phrases. Multimedia proxy-caches, Stream adaptation, Streaming protocols.

©2007 Babeş-Bolyai University, Cluj-Napoca

However, there are some situations when the proxy must further reduce its bandwidth usage. These situations can appear when the client's connection to the proxy is overloaded or when the proxy deals with heterogeneous clients (e.g. PDA devices, mobile phones, personal computers, etc.) and dynamically changing user preferences. In these cases, the proxy might choose to adapt the multimedia objects from its cache (instead of sending a new request to the origin server), so that the client is able to use the data it receives and the quality of this data is satisfactory.

The next section presents some other scenarios when adaptation might be useful in order to reduce the total amount of used bandwidth or meet client expectations. Section 3 provides a short overview of some of the techniques that can be used when doing adaptation. Following we present some of the available mechanisms inside streaming protocols that can assist both the client and the proxy-cache/server when deciding to perform an adaptation operation. A description of the test bed we used for the evaluation of those mechanisms as well as a discussion of the obtained results are presented in Section 5 and Section 6. Further away we present our conclusions and indicate possible directions for future work.

2. Use case scenarios

We have already mentioned the case of a multimedia proxy-cache having to service heterogeneous clients (PDAs, mobile phones, desktop computers, tablet PCs, etc.). In order to do this, the proxy-cache has multiple options: request the streams encoded in the appropriate format from the server (the adaptation is done by the server if an appropriate encoded version of the requested file is not available), perform the adaptation operation locally on already cached objects, or, if the object is not available locally and the server is not able to provide the requested quality, perform adaptation on the fly, on the streams it currently receives and delivers to active clients.

Most of the time the decision to perform adaptation is made at the proxy or at the server side before the start of the streaming session, but there are also situations when the client must take such a decision dynamically during the streaming session. During a video conference conference, each participant will receive and playback the streams from other participants and, at the same time, each participant streams the data captured (e.g. from a webcam) at the highest quality permitted by the available bandwidth. As not all the participants speak simultaneously, it would make sense that the client requests that the stream of the active speaker is delivered at full quality while the ones belonging to inactive participants are streamed with minimum quality. In such a case, the proxy could transcode the streams of the inactive clients and deliver them at a lower resolution and/or in grayscale. This would lower the bandwidth consumption while also reducing the burden of displaying all the streams in full quality for the clients with limited

or insufficient computational power and/or with limited display capabilities. Another example is when at some point during the playback of a music video, the user might decide to only listen to the audio track and disregard the video content while busy with another task. At that point he sends an adaptation request to the proxy/server in order to receive only the audio data. A similar scenario, when dynamic client-driven transcoding/adaptation is required, can be identified in a security surveillance system with surveillance cameras sending low quality streams, that switch to a higher quality when triggered by a sensor or by a human operator. Dynamic client-driven adaptation requests can also be used in a client-server/proxy environment as a coarser-granularity replacement for feedback information.

3. Short Survey of Adaptation Techniques

Adaptation of multimedia content means to be able to either enhance or reduce the quality of the data in concordance with the user preferences and the terminal capabilities it specifies.

When it comes to video data (video streams), the most common and frequent operation that is done is to reduce the quality of the video. In the following, we will refer only to situations in which the quality of a video stream is reduced by transcoding operations. By *video transcoding*, one understands the operation of converting a video from one format into another format, where a format is defined by characteristics such as bit-rate, frame-rate, spatial resolution, coding syntax and content. Transcoders can operate both in compressed and pixel-domain, the main difference being that, when operating in the compressed domain, the video data does not need to be decoded, transformed and then re-encoded (like in the pixel domain architectures). This leads to faster but limited (when it comes to the complexity of the operation) transcoding operations in the compressed domain.

In the following we will refer to three types of adaptation: temporal adaptation (which is done in the compressed domain), grayscale and size reduction (which are done in the decompressed or pixel domain). *Temporal adaptation* means dropping frames so that a lower average bitrate of the stream is achieved. *Size reduction* means down-sampling to a lower spatial resolution; the frame rate remains the same while the bitrate is reduced. *Grayscale reduction* drops the chrominance information (U and V in YUV format) obtaining a reduced bitrate (chrominance information makes up to 20% of a bitstream).

4. Dynamic Client-Driven Adaptation Support in Multimedia Streaming Protocols

When speaking about adaptation there are 3 things that have to be considered: who performs the operation, who decides when adaptation should be performed and the moment when the decision is taken.

In a client - server/proxy environment, adaptation can be performed both at the transmitter (i.e. server or proxy) and at the receiver side (i.e. client player). Generally, it is preferred that adaptation is performed at the transmitter's side, because of the following reasons: (a) multimedia adaptation like transcoding can be quite resource-consuming and usually the transmitter has greater computing power than the receiver and (b) an adapted multimedia stream usually has smaller demands for network bandwidth, so if the adaptation is performed at the transmitter, the network's bandwidth is used more efficiently.

The decision to adapt a multimedia stream is usually taken by the transmitter based on feedback from the client and on its current load. However, there are situations (see section 2) when it is necessary that the receiver takes the adaptation decision (even if the adaptation process itself is still carried out by the transmitter).

Regarding the moment when the decision to adapt is taken, this can either be (a) at the beginning of the streaming, during the session negotiation part (e.g. in the case of heterogenous clients, terminal capability negotiation, etc.), or (b) arbitrarily during the streaming session (e.g. session migration, changing user preferences, congested network links, subjective user decision, etc.).

Most of the standard streaming protocols provide little support for dynamic client-driven adaptation. SDP [1] includes partial support for terminal capabilities descriptions, while RTP/RTCP [2] supports sending feedback information (number of packets received/dropped, bandwidth received, etc.), from the client to the server, but provides no support for sending an adaptation decision from the client to the server. Extensions to RTSP [3] for stream switching allow the server to notify the client about stream switching. To our knowledge, none of these streaming protocols have strong support for allowing the client to communicate adaptation requests dynamically to the server. In the rest of this section, we briefly describe the Adaptation-aware Multimedia Streaming Protocol (AMSP), an experimental streaming protocol which provides support for dynamic client-driven adaptation.

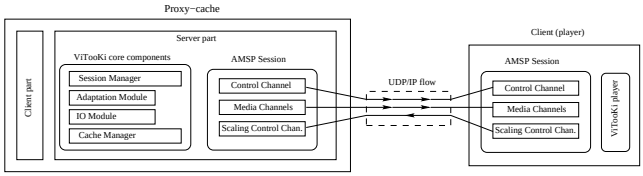


FIGURE 1. The streaming system's architecture

The Adaptation-aware Multimedia Streaming Protocol [4] is a streaming protocol similar to RTP. It conveys time sensitive information like multimedia data together with scaling information, so that multimedia streams can be adapted inside the network according to its rapid changing parameters. The scaling information can be used by common core routers to perform packet-level adaptation

of multimedia streams (i.e. drop less important packets) or by scaling proxies to perform complex media transformations inside the network (e.g. color reduction, temporal reduction, transcoding, etc.). The central concept of AMSP is the channel concept. Each channel is identified by an 8-bit field in the AMSP header called the ChannelID field. There are several types of channels AMSP supports: *control channel* (conveys configuration information global to the AMSP session), *media channels* (deliver multimedia data), *metadata channels* (transport metadata), *scaling control channels* (transport scaling information), *retransmission channels*, *feedback channels* and *auxiliary channels* (transport other types of information). A multimedia stream is mapped onto one or more media channels, and thus, its packets get the priority of the respective channel(s). Basically, an AMSP session is a multiplexation of several AMSP channels (at least the control channel). The Scaling Control Channel of an AMSP session can be used to convey scaling hints or adaptation requests to multimedia proxys or streaming servers to steer the adaptation of multimedia streams according to the network's load or the client's decision. One or more scaling hints are encapsulated in a scaling control channel packet which contains the AMSP header (including the channel id of the packet) and one or several scaling rules. Each scaling rule contains the scaling action (type of adaptation) to be performed (e.g. temporal reduction, size reduction, color reduction, requantization, etc.), the media channels on which this adaptation should be applied, quality and size reduction expected if this scaling rule is applied and payload specific to each rule.

We have evaluated the use of AMSP scaling hints and AMSP scaling control channel to enable dynamic client-driven adaptation decision support in multimedia proxy-caches. The architecture of our streaming environment is described in the following section.

5. The streaming environment used for evaluation

The streaming environment is built around the ViTooKi framework [5]. ViTooKi (The Video ToolKit) is an open source, high-level C++ multimedia library created with the goal of simplifying the implementation of multimedia applications. It offers encoding/decoding support for a number of video and audio formats, streaming via RTP/UDP, various adaptation techniques, meta-data support (MPEG-7 and MPEG-21), session management through RTSP and SDP and also cache management. It also includes some useful applications like a streaming server and a player.

The streaming environment used in our experiments is depicted in Figure 1. We used the AMSP library developed in [4] and integrated AMSP in ViTooKi as an IO streaming class so that we can stream multimedia data using AMSP. We also implemented the scaling control channel which was not implemented in the original AMSP library. The proxy architecture includes ViTooKi core components for managing multimedia sessions and for managing the cache, the adaptation

components for performing adaptation and the IO module for reading and decoding the multimedia content. All these ViTooKi components are not directly related to streaming. For streaming multimedia data the proxy uses an AMSP session which contains three kinds of channels: one control channel (necessary in an AMSP session for configuring the other channels), several media channels for sending multimedia data and a scaling control channel for receiving scaling hints (adaptation requests) from the client.

The experiments presented in the following section show the effects of using AMSP scaling hints on the state of the system (client, proxy and network) using the following metrics: client's perceived quality of multimedia data and the network bandwidth used. We make the following remark on the experiments: due to the fact that we want to show the effects of enabling dynamic client-driven adaptation decisions in multimedia proxy-caches on the network bandwidth used and on the client's perceived quality, we assumed multimedia data was always present in the proxy's cache so that the proxy doesn't have to get the data from the originating streaming server and moreover we ignored all the overhead related to the management of cache objects which is the same as when AMSP scaling hints were not used.

6. Experiments and evaluation

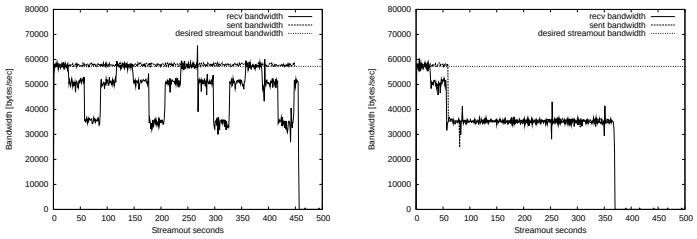
To evaluate the effects of enabling dynamic client-driven adaptation decisions through AMSP scaling hints in multimedia proxy-caches we streamed an MPEG-4 video from the proxy to the client (as depicted in Figure 1), through a traffic-shaped network line. We used a Linux traffic shaper on the network link between the proxy and the client and this traffic shaper changes the available bandwidth every 30 seconds using the following pattern: 75KB/s - 60KB/s - 36.7KB/s - 60KB/s - 75KB/s

So, in the first 30 seconds of streaming, the available bandwidth is 75KB/s, then it drops to 60KB/s, then after another 30 seconds it drops down to 36.7KB/s and at this point, after another 30 seconds, the available bandwidth climbs back to 75KB/s in two 30 seconds long steps. The above pattern is repeated indefinitely. The video stream used in our tests was the MPEG-4 reference stream "Big Show One + Two" with 13,000 frames and a frame rate of 25 fps in CIF resolution. The average bitrate of the stream is 400 kbps, with quantization levels of 28 for B-VOPs and 16 for I-VOPs and P-VOPs. The stream was encoded with the following frame pattern in each one second GOP: IBBBBPBBBBPBBBBPBBBBPBBBB. In order to avoid client buffer underruns, we considered the desired streamout rate of 457.7 kbps (57.2KB/s) which is a little higher than the average bitrate of the stream.

We performed three experiments and in each run we measure the effects on the network bandwidth used and on the client's perceived quality which we measure using PSNR. In the first run, the video is not adapted and it is streamed constantly

at the desired streamout rate of 57.2KB/s no matter what the available bandwidth is. In the second run, in the beginning, the proxy streams the video unadapted, at a constant streamout rate of 57.2KB/s, but around second 65, after the available bandwidth drops to 36.7KB/s, the client sends a color adaptation request through the AMSP scaling control channel and the proxy adapts the video at an average bitrate of 35KB/s using color to grayscale reduction. The third run is exactly like the second one, only that at second 65, temporal adaptation is used to adapt the video stream at 35KB/s.

By employing the aforementioned bandwidth fluctuations and by making the client request adaptation after the available bandwidth drops to 36.7KB/s, we think that our experiments emulate both use cases presented in section 2, i.e. when the client decides to request adaptation based on the degradation of the network conditions (i.e. AMSP scaling hints are used as a coarser-granularity replacement for feedback) and when the client decides to request adaptation based on its own subjective desires.



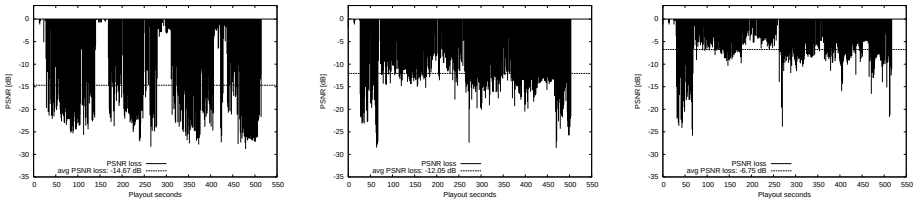
(a) for the unadapted video stream (b) for the color adapted video stream

FIGURE 2. Bandwidth evolution

For the first streaming scenario, when the video is streamed unadapted at a constant streamout rate of 57.2KB/s, Figure 2(a) shows the evolution of the proxy's sent bandwidth and the bandwidth received by the client. We see that although the proxy streams the video at the desired streamout rate of 57.2KB/s, the client receives data at a bandwidth that follows the fluctuations imposed by the traffic shaper. In this streaming scenario, according to Figure 3(a) the stream received by the client loses up to 27 dB PSNR due to lost I-, P- and B-VOPs. The average quality reduction is 14.67 dB.

For the second experiment, although the proxy starts streaming the video at the desired streamout rate (i.e., 57.2KB/s), after 65 seconds, due to a client adaptation request received through the AMSP scaling control channel, the video stream is adapted at a bitrate of 35KB/s using color to grayscale adaptation. The evolution of the sent and received bandwidth is shown in Figure 2(b). It can be seen, that

after second 65, the sent and received bandwidth equal 35KB/s. According to Figure 3(b) the quality degrades with an average of 12.05 dB PSNR due to lost VOPs. We note that, after color adaptation was applied, the quality reductions are less severe than the ones obtained in the first experiment. Please note that in the first 65 seconds, the PSNR loss is the same for all three experiments.



(a) for the unadapted video stream (b) for the color adapted video stream (c) for the temporal adapted video stream

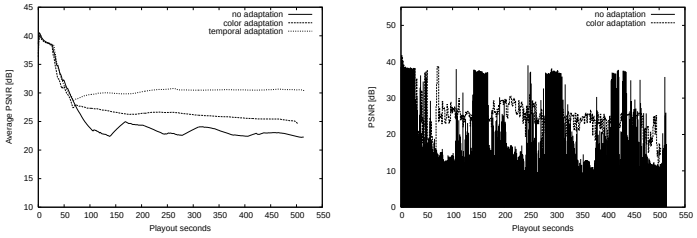
FIGURE 3. Quality loss

The bandwidth evolution for the third experiment, when temporal adaptation is applied on the stream after second 65, is the same as the one depicted in Figure 2(b). According to Figure 3(c) the average quality reduction is 6.75 dB which is less than the quality reductions observed in the first two experiments. This fact is shown more clearly in Figure 4(a) where the average PSNR values obtained for all three experiments are plotted as a function of playout seconds. We can see there, that in the first 65 seconds, the average PSNR values are the same for all three experiments, but after the client decides to request adaptation from the proxy, the temporal adaptation scenario achieves the greater PSNR average, followed by the color adaptation scenario and last, by the scenario when the video was not adapted at all.

We also note that, when the available bandwidth is high (i.e., 75KB/s), the unadapted stream achieves the greatest PSNR values, so the greatest quality. This is shown in Figure 4(b) where the PSNR values obtained for the unadapted stream and the ones obtained for the color adapted stream are compared.

7. Conclusion

We have evaluated the use of dynamic client-driven adaptation decision support in multimedia proxy-caches through the use of AMSP scaling control channel and AMSP scaling hints. Our experiments show that providing dynamic client-driven adaptation support has its benefits on the streaming environment and in some scenarios, it is the only viable solution.



(a) Average PSNR comparison for the three experiments

(b) Comparison of PSNR values for the unadapted and color adapted video stream

FIGURE 4. PSNR comparisons

REFERENCES

[1] M. Handley, V. Jacobson, SDP: Session Description Protocol, RFC 2327, April 1998.
[2] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC 3550, July 2003.
[3] H. Schulzrinne, A. Rao, R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326, April 1998.
[4] M. Ohlenroth, Network-Based Adaptation of Multimedia Contents, PhD. Dissertation, University of Klagenfurt, Austria, September 2003.
[5] The ViTooKi Framework, http://vitooki.sourceforge.net/

"BABES-BOLYAI" UNIVERSITY, CLUJ NAPOCA
E-mail address: {forest, claudiu, florin, bufny}@cs.ubbcluj.ro