Optimized adaptation of video streams in streaming servers

Adrian Sterca
Babes-Bolyai University
Computer Science Department
Cluj-Napoca, Romania
forest@cs.ubbcluj.ro

Abstract

In this paper we investigate stream adaptation policies for video streaming servers which use smooth TCP-friendly congestion controls to compute the sending bitrate. We seek an optimal adaptation policy which maximizes the perceived quality at the client side and tries to avoid an empty prefetch buffer at the client side. In order to do so, we first develop an optimization framework for our problem and then suggest an algorithm to solve it.

1. Introduction

Multimedia streaming applications are on an ascending curve regarding Internet bandwidth usage nowadays. Because a best-effort network like the Internet does not guarantee a constant level of service, streaming applications must continuously adapt to changing QoS parameters of the network. This operation is two-fold: first, the sending bitrate must meet the available bandwidth and secondly, the encoded stream must support the aforementioned bitrate adaptation.

In order to minimize quality variations among consecutive scenes at the receiver, streaming applications often use smooth TCP-friendly congestion control algorithms like TFRC [1] [2] and binomial congestion controls [3] which give more or less stable QoS conditions in the Internet. This smooth variation of available bandwidth achieved by TCP-friendly congestion controls is better supported than the saw-tooth variation obtained by TCP's AIMD mechanism by modern audio-video codecs through stream adaptation techniques like: FPS (i.e. Frames per Second) reduction, color to gray-scale reduction, size reduction etc. However, given a relatively stable bandwidth like the one given by a TCP-friendly congestion control, applying these stream adaptation techniques on the original stream in order to meet this specific bandwidth remains a challenge because it must maximize the perceived quality of the stream in the given network conditions and also it must avoid a buffer underrun at the client side.

In this paper we address the problem of adapting a layered scalable video stream to a network bandwidth which is given by a smooth TCP-friendly congestion control and is assumed to vary slowly throughout the streaming session. The goal is to derive a level of stream adaptation that avoids buffer underrun at the client and achieves the maximum perceived quality at the client in the given network conditions.

The contribution of this paper is an optimization framework for dealing with the previous mentioned problem. The paper is organized as follows. In Section 2 we visit some related work. Then Section 3 describes our optimization framework and outlines an algorithm to solve it. Section 4 presents some preliminary simulations and Section 5 deals with implementation issues of our algorithm. The paper ends with conclusions and future work ideas.

2. Related work

Traditionally, during streaming, the level of stream adaptation is set using threshold values for the player's buffer. When the player's buffer drops under a low threshold value, the stream bitrate reduction level increases substantially and if the buffer rises above a high threshold value, the reduction level decreases [4]. In [5] an optimal stochastic control problem is formulated for choosing the level of stream adaptation and a heuristic threshold policy is devised for solving this problem. However, avoiding a zero prefetching buffer at the client side does not seem to be a real issue here. De Cuetos and Ross develop in [6] a real-time heuristic for adapting the bitrate of a FGS stream to the throughput of a TCP or TFRC flow and their solution fairly tracks an optimum value which maximizes the perceived quality. Our framework differs from the ones above in that it uses classic optimization theory and does not imply the use of any thresholds.

3. The optimization framework

We consider the process of streaming a video from a server to a client over a best-effort network. The client starts prefetching and after the prefetching buffer reaches a certain level playback of the movie is started.

3.1. Notations and model

We use the following notations in the rest of the paper:

n	number of playing seconds in the video stream
sec	number of current streaming second
X	average bandwidth from the start of streaming until now
Δ	current size of prefetching buffer (in seconds)
b	the bitrate of the current streaming second (bytes)
b_{avg}	the average bitrate of the whole stream (bytes/sec)
a	the level of adaptation (reduction) of the stream in the current second ($a \in (0,1]$)

Occasionally, we will also use subscript variants of the above notations, e.g. a_i for the level of adaptation of the i-th second from the stream and b_i for the bitrate of the i-th second from the stream.

While estimating available bandwidth is the job of a congestion controller and should be done at a fine granularity, adapting the stream is the job of the application and should be done at a coarser granularity in order to avoid too much quality variability between consecutive frames. Generally, the available bandwidth is updated on a shorter timescale than adaptation level a which is updated at the beginning of each streaming second and remains unchanged for the duration of that second. The level of adaptation a_i is a real number in the interval (0,1] and represents the fraction of the stream's bitrate in second i (i.e. b_i) which is sent to the client. The rest of the stream second i, i.e. $(1-a_i)b_i$ bytes, is discarded at the sender. Which exactly $(1-a_i)b_i$ bytes are discarded from the i-th stream second depends on the adaptation technique being used. For example, if fps reduction is being used on a stream with 30 frames per second and $a = 1/4$, exactly k frames are discarded from the current second if and only if the sum of those k frames' sizes is approximately $1/4$ from the total bitrate of the stream second. In this paper, we primarily focus on fps reduction, but other adaptation techniques could be used as well.

If $a = 0$ then the current stream second is not sent to client at all, but it is discarded entirely at the server side. When $a = 1$ the current stream second is sent to the client as it is without adapting it.

3.2. Optimization framework

We want to model an optimization framework that would help us develop a stream adaptation policy which maximizes the perceived quality and maintains a non empty buffer at the client. In order to do so we seek a utility function which maximizes benefit (i.e. perceived quality) and minimizes cost (i.e. low prefetch buffer value). Consequently, our optimization problem is the following:

$$\begin{cases} maxU(a) = q(a) - C(a) \\ a \in (0,1] \end{cases} \quad (1)$$

where $q(a)$ is perceived quality function and $C(a)$ is the cost for choosing an adaptation level equal to a.

Benefit

We model the quality benefit using Rate-Distortion functions. In classical rate-distortion theory one popular statistical model for DCT compressed data is that of a Gaussian memoryless source with mean $\mu = 0$ and variance σ^2, for which we have the following rate-distortion formula [7]:

$$D(R) = \sigma^2 2^{-2R} \quad (2)$$

where $D(R)$ is the distortion obtained when the encoding rate is R. When distortion is D, the quality obtained (i.e. benefit) is measured using the PSNR formula:

$$PSNR = 10log_{10}(\frac{max^2}{D})$$

where max is the maximum amplitude of the signal and D is the distortion. We note that instead of equation (2) the rate-distortion function from [8] could also be used.
Using the previously described formulas, we now can specify the quality benefit of a certain level of adaptation $a \in (0,1]$:

$$\begin{array}{l} q(a) = 10log_{10}(\frac{b_{max}^2}{D(ab)}) \\ where \quad D(ab) = \sigma^2 2^{-2ab} \end{array} \quad (3)$$

where b_{max} is the maximum bitrate per second from the stream, b is the bitrate for the current second of the stream and σ^2 is the variance of the stream's bitrate. After performing some simple computations in equation (3), we can write the benefit function like this:

$$q(a) = 10log_{10}(\frac{b_{max}^2}{\sigma^2}) + 20\,a\,b\,log_{10}2 \quad (4)$$

You can see in equation (4) that when a decreases, the quality function also decreases.

Cost

The cost implied in the process of adaptation is reflected in the value of client's prefetch buffer which increases at a lower speed (even negative) as the level of adaptation a gets closer to 1. More specifically, if playing starts with an initial buffer value Δ_0, the evolution of the buffer during the streaming session is like this (assuming X is not too much bigger than the average bitrate of the stream):

$$\Delta_1 = \Delta_0 + \frac{X}{a_0 b_0} - 1$$
$$\Delta_2 = \Delta_1 + \frac{X}{a_1 b_1} - 1$$
$$...$$
$$\Delta_i = \Delta_{i-1} + \frac{X}{a_i b_i} - 1$$

We choose our cost function so that it obeys the following rules:

- $C(a) \geq 0$
- if a increases the value of the buffer, than the cost should decrease
- if a decrease the value of the buffer, than the cost should increase
- the cost function must be scaled with $1/\Delta$ and also with b_{avg}

The following function obeys the aforementioned rules and is a good candidate for our cost function:

$$C(a) = k\left(e^{4(1-\frac{X}{ab})} + 2e^{4(1-\frac{X}{ab}-\frac{b}{(n-sec)\,b_{avg}}\Delta)}\right)\frac{b_{avg}}{\Delta} \quad (5)$$

where $k > 0$ is a scaling factor which should scale the cost function with the benefit function.

3.3. Algorithm for solving problem (1)

In order to solve the problem (1), we first obtain the first-order and second-order derivatives of $U(a)$:

$$\frac{\partial U}{\partial a} = 20b\,log_{10}2 - k\frac{b_{avg}}{\Delta}\left[e^{4(1-\frac{X}{ab})}\frac{4X}{a^2b}\right] + 2k\left[e^{4(1-\frac{X}{ab}-\frac{b}{(n-sec)\,b_{avg}}\Delta)}\frac{4X}{a^2b}\right]\frac{b_{avg}}{\Delta}$$

$$\frac{\partial^2 U}{\partial a^2} = ke^{4(1-\frac{X}{ab})}\left[\frac{8X}{a^3b} - \left(\frac{4X}{a^2b}\right)^2\right]\frac{b_{avg}}{\Delta} + 2ke^{4(1-\frac{X}{ab}-\frac{b}{(n-sec)\,b_{avg}}\Delta)}\left[\frac{8X}{a^3b} - \left(\frac{4X}{a^2b}\right)^2\right]\frac{b_{avg}}{\Delta}$$

In the following lines, we use the value of $k = 1$ for the scaling factor. Next, we observe that $\frac{\partial U}{\partial a}(0) > 0$ and if $1 < \frac{2X}{a\,b}$ then function $\frac{\partial^2 U}{\partial a^2}(a) < 0$ (i.e. $U(a)$ is strictly concave). With these in mind we can use the following algorithm to solve problem (1):

1) If $\frac{2X}{b} \geq 1$ *then:*
Compute $\frac{\partial U}{\partial a}(1)$*;*
1.1) If $\frac{\partial U}{\partial a}(1) > 0$*, then since* $\frac{\partial U}{\partial a}(0) > 0$ *and* $\frac{\partial U}{\partial a}(a)$ *is continuous on (0,1], we can conclude that* $U(a)$ *is increasing on (0,1] and* $a = 1$ *is the solution to problem (1);*
1.2) Else, find the maximum of $U(a)$ *using a gradient ascend algorithm [9]; we know a maximum exists since* $U(a)$ *is concave if* $\frac{2X}{b} \geq 1$*;*
2) If $0 < \frac{2X}{b} < 1$ *then:*
$U(a)$ *is convex on* $[\frac{2X}{b}, 1]$ *(i.e.* $\frac{\partial U}{\partial a}(a)$ *is increasing) and* $U(a)$ *is concave on* $(0, \frac{2X}{b})$ *(i.e.* $\frac{\partial U}{\partial a}(a)$ *is decreasing). In this case we use the same gradient ascend algorithm we used at 1.2) to find the maximum.*

4. Preliminary Simulations

We present in this section some preliminary simulations of the optimization framework we have outlined in the previous sections. In our simulations we have used an MPEG-4 video with the following one second bitrate values:

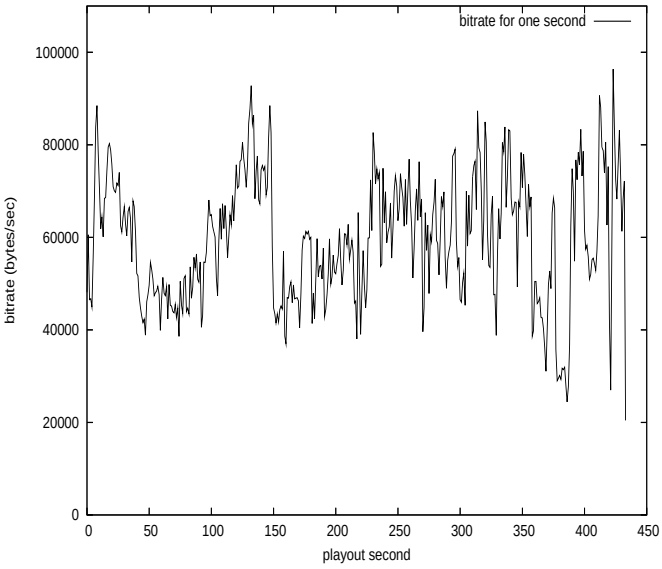

Figure 1. The 1 second bitrates of the stream

We have used this video because it has a reasonable number of scene changes, so that the bitrate is very variable. A typical shape of our utility function is depicted in Fig. 2. The function was computed for the following values: $X = 35000\,bytes/sec, b_{avg} = 57915, n = 433, sec = 100, \Delta = 10$ and b has bitrate values ranging from 20438 to 96365.

The adaptation levels computed by our algorithm for the following values: $X = 50000\,bytes/sec, b_{avg} = 57915, n = 433, \Delta = 10$ are depicted in Fig. 3.

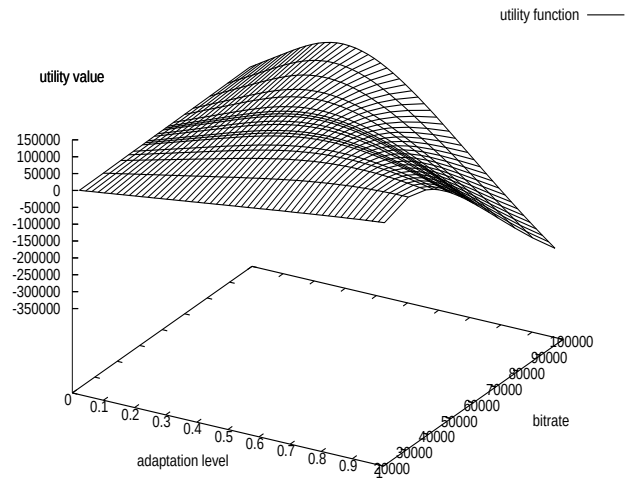

Figure 2. Typical shape of the utility function

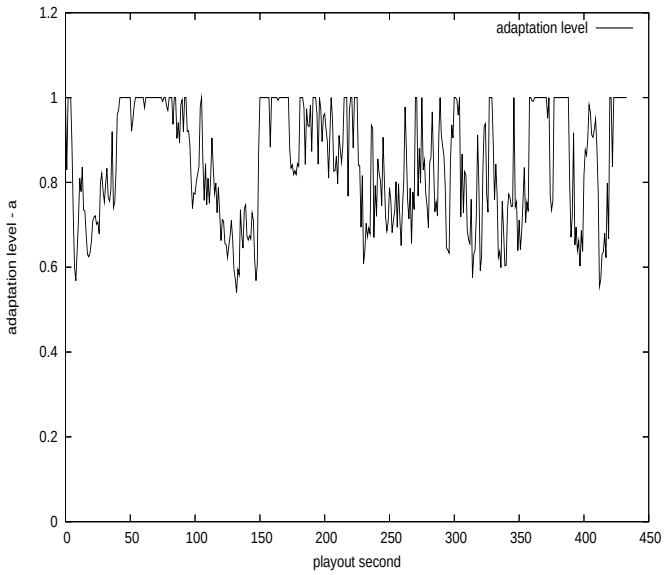

Figure 3. Adaptation levels computed by our algorithm

5. Implementation issues

In this section we discuss some implementation issues related to our algorithm. When solving problem (1) using a gradient descend algorithm we do not need to obtain an exact optimum since at least for fps reduction we can not drop from the stream a number of bytes which is less than the size of the smallest frame from the current second, so an error of 0.01 from the exact optimum is quite reasonable.

Also, during our simulations we have observed that a number of maximum 10 iterations of the gradient ascend algorithm are quite enough for solving problem (1) with the approximation error of 0.01.

6. Conclusions

In this paper we presented an optimization framework and an algorithm to determine an optimal adaptation policy for video streaming servers which maximizes the perceived quality at the client side and tries to avoid an empty prefetch buffer at the client side. The next step is to implement this adaptation policy in a real streaming server and see how it behaves in a test environment.

References

[1] S. Floyd, M. Handley, J. Padhye, J. Widmer, *Equation-Based Congestion Control for Unicast Applications*, ACM SIGCOMM 2000.

[2] S. Floyd, M. Handley, J. Padhye, J. Widmer, *TCP Friendly Rate Control*, RFC 3448, January 2003.

[3] D. Bansal, H. Balakrishnan, *Binomial Congestion Control Algorithms*, IEEE Infocom 2001.

[4] T. Phelan, *Strategies for Streaming Media Applications Using TCP-Friendly Rate Control*, Internet Draft, draft-ietf-dccp-tfrc-media-00.txt, July 2005.

[5] D. Saparilla, K. W. Ross, *Optimal Streaming of Layered Video*, IEEE Infocom 2000, vol. 2, pp. 737-746, Tel Aviv.

[6] P. de Cuetos, K. W. Ross, *Adaptive Rate Control for Streaming Stored Fine-Grained Scalable Video*, The 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002), Miami, USA, pp. 3-12, 2002.

[7] T. M. Cover, J. A. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[8] M. Dai, D. Loguinov, *Analysis of Rate-Distortion Functions and Congestion Control in Scalable Internet Streaming*, The 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2003), California, USA, 2003.

[9] D. Bertsekas, *Nonlinear Programming*, Athena Scientific, ISBN 1-886529-00-0, April 2004.