Choosing the level of stream adaptation in TCP-friendly congestion controlled video 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 transmission rate. TCP-friendly congestion controls compute, by specific means, the rate at which the streaming server can send video data (i.e. available bandwidth in the network). Of course, this rate must be supported at application level by an appropriate stream adaptation mechanism. 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 the given TCP-friendly conditions. We formulate our problem in an optimization framework and then devise a gradient ascend algorithm to solve it. The solution is an adaptation policy that can be used to support TCP-friendly rate control at codec (application) level.

1. Introduction

Nowadays, multimedia streaming applications make for an increasing percentage of the data transferred over the Internet. However, the heterogeneity and the best-effort nature of the Internet pose great challenges to multimedia streaming applications. 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 transmission rate must meet the available bandwidth and secondly, the encoded stream must support the chosen transmission rate.

In order to minimize quality variations among consecutive scenes at the receiver, streaming applications often use smooth TCP-friendly congestion control algorithms like TFRC (*TCP-Friendly Rate Control*) [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, there is still a trade-off between achieving a maximum stream quality at the receiver and maintaining a reasonable sized prefetch buffer at the receiver to avoid movie freezing due to an empty buffer. Applying the aforementioned stream adaptation techniques on the original stream in order to meet a specific TCP-friendly bandwidth remains a challenge because it must deal with this trade-off. In other words, a stream adaptation policy must maximize the perceived quality of the stream in the given network conditions (which decreases buffer size in case of insufficient bandwidth) and also it must avoid a buffer underrun at the client side.

In this paper we address the problem of adapting a layered scalable, DCT-based (*Discrete Cosine Transform*), video stream to a transmission rate computed by a smooth TCP-friendly congestion control and which 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, thus, a policy for choosing an optimal stream adaptation level (with respect to an utility function) for TCP-friendly computed bandwidth. The paper is organized as follows. In Section 2 we visit related work. Then Section 3 describes our optimization problem and a solution to this problem is given in Section 4. Section 5 presents experiments that depict the characteristics of our adaptation policy and Section 6 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 prefetch 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 sized prefetch 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.

In [10] authors integrate the stream adaptation policy into transport-level congestion control and they develop a congestion control based on TFRC which considers media characteristics, but their framework (especially the stream adaptation process) is focused on MPEG FGS streams, while ours is useful for general, DCT-based, layered scalable video streams.

3. The optimization framework

We consider the process of streaming a pre-recorded or real-time encoded video from a server to a client over a best-effort network like the Internet. The client starts prefetching and after the prefetch buffer reaches a certain level, playback of the movie is started. While estimating available bandwidth is the job of a congestion control implemented at the server and should be done at a fine granularity (typically once per round-trip time), 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. In our streaming environment, the available bandwidth is computed by TFRC based on client feedback once per round-trip time or when a packet loss occurs. In the beginning of each streaming second, the current level of stream adaptation is computed based on the available bandwidth estimated by TFRC and this adaptation level is applied on the stream, for the duration of that second.

We use the following notations in our paper:

n	number of playing seconds in the video stream
sec	number of current streaming second
X	transmission rate given by TFRC
Δ	current size of the client prefetch 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.

The adaptation level a 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. $a_i\, b_i$) 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. Of course, for fps reduction, B-frames must be dropped first from a GOP (*Group Of Pictures*) in a timely uniform way [11] and I- or P-frames should only be dropped from a GOP if all B-frames from that GOP were previously discarded.

If $a = 0$ then the current stream second is not sent to the 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.

Optimization problem

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 function

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]:

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

where $D(R)$ is the distortion obtained when the encoding rate is R. We use this model for formulating our benefit function because of its mathematical tractability, although it is only an approximation (not the best one) of the actual rate-distortion function of a DCT-based video stream (the actual rate-distortion function may not be continuous). Because DCT concentrates much of the entropy of the signal in the DC-coefficient and after quantization many of the AC-coefficients become 0, the distribution of DCT coefficients obtained can be modeled as a normal distribution with mean 0. Although function (2) is only an approximation of the actual rate-distortion function of the stream, it is good enough for us, as we associate a higher importance to the cost function which evaluates the possibility of the prefetch buffer to become empty.

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 power (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 $a \in (0, 1]$:

$$\begin{aligned} &q(a) = 10log_{10}(\tfrac{b^2}{D(ab)}) \\ &where \quad D(ab) = \sigma^2 2^{-2ab} \end{aligned} \tag{3}$$

where b is the bitrate for the current second of the stream and σ^2 is the variance of the stream's bitrate. We note that we have used b as the maximum power of the signal. This is not entirely accurate for the PSNR formula, but we can assume that the maximum power of the current stream second is achieved when the stream second is not adapted (i.e. $a = 1$, bitrate of the current second remains b). After performing some simple computations in equation (3), we can write the benefit function like this:

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

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

Cost function

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 at the beginning of the current streaming second, the client buffer size is Δ_i, the bitrate of the current stream second is b_i bytes, the available bandwidth computed most recently by TFRC is X and the value of the adaptation level for this second is a_i, then in the beginning of the next streaming second, the new buffer size can be approximated by this relation:

$$\Delta_{i+1} = \Delta_i + \frac{X}{a_i\,b_i} - 1 \tag{5}$$

because 1 second is consumed from the buffer and the buffer gets $a_i b_i$ bytes in the current second which is approximately $\frac{X}{a_i\,b_i}$ seconds of the stream. This means, that if $\frac{X}{a_i\,b_i} - 1 > 0$ after the current streaming session has passed, an adaptation level of a_i has increased the size of the buffer. Conversely, if $\frac{X}{a_i\,b_i} - 1 < 0$ the buffer size has decreased. Of course, in one second of time, the buffer can not decrease more then 1 stream second because only 1 stream second gets played in 1 second of time.

We seek a cost function so that obeys the following rules:

- $C(a) \geq 0$
- if a increases the value of the buffer, then the cost should decrease
- if a decreases the value of the buffer, then the cost should increase
- the cost function must be scaled with $1/\Delta$ and also with b_{avg}, meaning that if the buffer is large, increasing or decreasing such a buffer must have a small effect on the cost function (i.e. cost function should have a small value); also, cost must be scaled with b_{avg} to have comparable order of magnitude with the benefit function (which is $o(b)$).

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

$$C(a) = k\left(e^{l(1-\frac{X}{ab})} + m * e^{l(1-\frac{X}{ab} - \frac{b}{(n-sec)\,b_{avg}}\Delta)}\right)\frac{b_{avg}}{\Delta} \tag{6}$$

where $k > 0$ is a scaling factor which should scale the cost function with the benefit function, $l > 0$ is a general exponential scaling factor and $m > 0$ is the second exponential's scaling factor. The first term of the cost function, $e^{l(1-\frac{X}{ab})}$ exponentially penalizes the cost of an adaptation level, a, which decreases the size of the buffer (i.e. condition $\frac{X}{a_i\,b_i} - 1 < 0$). The second exponential term of the cost function has the role to further penalize the adaptation level if the decrease of buffer size is greater than $\frac{b}{(n-sec)\,b_{avg}}\Delta$.

By using this last value we wanted to asses the impact of the current choice for the adaptation level on the buffer size for the remaining $n - sec$ streaming seconds. We considered that, if available bandwidth X does not change (this is the reason we are using TFRC - because it gives a relatively constant throughput), the current stream second has the right "to eat" as much as $\frac{1}{n-sec}\frac{b}{b_{avg}}$ from the buffer Δ. If "it eats" more than this, the cost function must be penalized.

Throughout our paper and experiments, we use the following values for our scaling factors: $k = 1$, $l = 4$ and $m = 2$.

4. Optimal stream adaptation policy

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

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

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

Note that in the following lines we use specific values for k, l, m (i.e. $k = 1$, $l = 4$ and $m = 2$). 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.*

The gradient ascend algorithm we use for computing the maximum at 1.2) and 2) is a simple feasible method algorithm which starts at $a^0 = 1$ and in each iteration moves towards the optimum, using the following formula:

$$a^{k+1} = a^k + \beta^k(\overline{a}^k - a^k)$$

where $\beta^k \in (0, 1]$ is the stepsize and $\overline{a}^k$ is a feasible value obtained as the solution to the following optimization problem:

$$\begin{cases} max\,\frac{\partial U}{\partial a}(a^k)(a - a^k) \\ a \in (0, 1] \end{cases}$$

The solution of the previous maximization problem can be computed directly and it is $a = 1$ if $\frac{\partial U}{\partial a}(a^k) \geq 0$ and a value close to 0 otherwise. By choosing $\overline{a}^k$ this way, we are sure we are moving in the right direction (i.e. in the direction in which the value of U is increased; $\frac{\partial U}{\partial a}(a)(a_{k+1} - a_k) > 0$). It must be noted here, that 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.

In our experiments we used an initial value of $\beta^0 = 0.2$ for the stepsize and β was decreased each iteration according to $\beta^{k+1} = \beta^k/1.05$.

5. Experiments

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

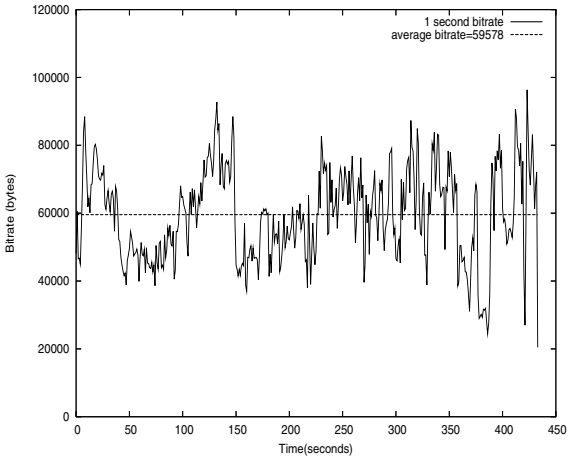

Figure 1. The 1 second bitrate of the stream

The average bitrate of the stream is 59578 bytes/second.

We implemented our adaptation policy in the ViTooKi streaming framework [12]. We streamed the video presented in Fig. 1 from our video server to the video client. The server and the client ran on two separate Pentium IV 2GHz computers with linux-2.6.16 which are linked by a traffic-shaped router. There were also 8 TCP flows sharing the bandwidth shaped by this router. We used two network setups. In the first setup the router sets the bandwidth to

600 KB/sec and in the second one the bandwidth is set to 540 KB/s. Because our TFRC implementation in ViTooKi is at application level, while TCP is implemented in the kernel of the operating system, TFRC was less efficient than TCP in using available bandwidth, so the multimedia flow received slightly less bandwidth than the TCP flows. The transmission rate achieved by the multimedia flow in the first setup is depicted in Fig. 2 where we can see that the average transmission rate for the whole streaming session was 43752 bytes/sec. For the second network setup the average transmission rate achieved by the multimedia flow was 38003 bytes/sec (Fig. 3). The adaptation levels computed by our stream adaptation policy together with the values of the buffer size in both network setups can be seen in Fig. 4 and Fig. 5, respectively.

We can see that for the first network setup the stream adaptation required to maintain a non-empty buffer was small (i.e. $a = 1$ most of the time, so no adaptation was performed) even though the average transmission rate of TFRC was smaller than the average bitrate of the stream (i.e. 43752 bytes/sec). But for the second network setup adaptation levels required to maintain a non-empty buffer were higher because the average available bandwidth was only 38003 bytes/sec.

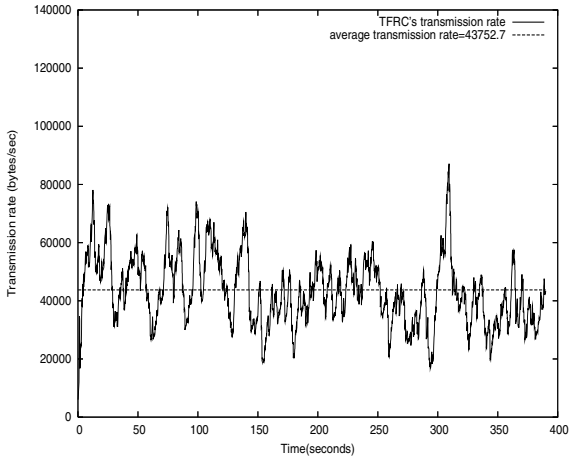

Figure 2. The transmission rate evolution for the 1st setup

6. Implementation issues

In this section we discuss some implementation issues related to our algorithm. When solving problem (1) using a gradient ascend 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

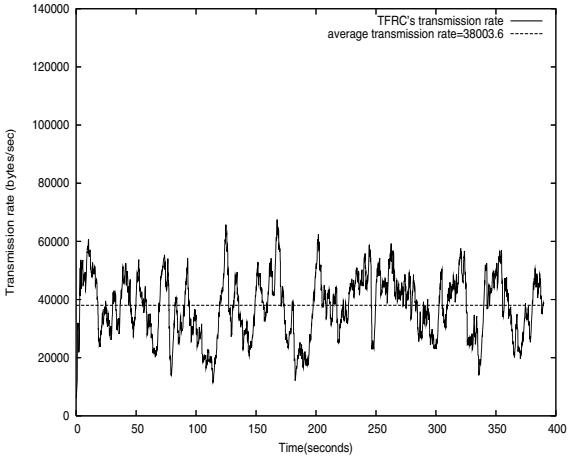

Figure 3. The transmission rate evolution for the 2nd setup

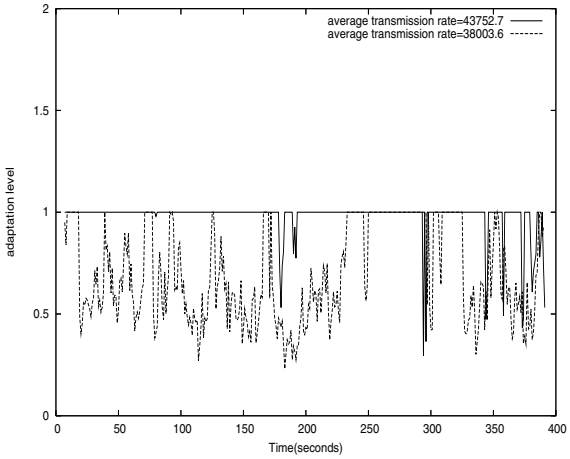

Figure 4. The evolution of adaptation level a

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 experiments 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.

7. Conclusions

In this paper we have presented an optimization framework and an algorithm to determine optimal adaptation policy for video streaming servers using TCP-friendly congestion control which maximizes the perceived quality at the client side and tries to avoid an empty prefetch buffer at the client side. Experiments showing that this optimal adaptation policy maintains a non-empty buffer in different net-

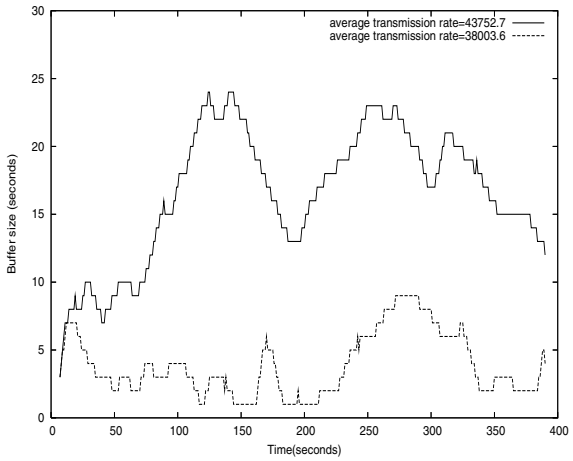

Figure 5. The evolution of the buffer size

work setups were presented. We need however to perform more tests and to measure also the quality perceived by the client.

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.

[10] J. Yan, K. Katrinis, M. May, B. Plattner, *Media- and TCP-Friendly Congestion Control for Scalable Video Streams* IEEE Transactions on Multimedia, Vol. 8, No. 2, April, 2006.

[11] M. Kropfberger, *Multimedia Streaming over Best-Effort Networks using Multi-Level Adaptation and Buffer Smoothing Algorithms*, PhD thesis, University of Klagenfurt, October, 2004.

[12] The ViTooKi Project, http://vitooki.sourceforge.net/.