Distortion-optimized Rate Shaping for TCP-friendly Video Streaming

Imed Bouazizi & Stephan Wenger
Nokia Research Center
Visiokatu 1, Tampere 33720, Finland
E-mail: imed.bouazizi@nokia.com & stewe@stewe.org

***Abstract*— In this paper, we propose a rate control algorithm, which determines the optimal packet scheduling for transmission, in order to minimize the overall expected distortion. We introduce a distortion estimation framework, which we use to estimate the damage that the loss of each single video packet will cause on the video quality. The packet distortion is estimated separately for base-layer and enhancement layer packets. We then present an algorithm to estimate the available rate of the channel based on RTCP feedback information. Finally, an algorithm to determine which packets to discard, in order to minimize the overall distortion while adhering to the estimated rate is described. We evaluate our algorithm through simulations using FGS layered MPEG-4 video sequences and show the significant improvements in video quality that can be achieved using distortion-based rate shaping.**

***Index Terms*— rate shaping, MPEG-4, rate control, FGS, video streaming**

I. Introduction

Packet loss and excessive delays are mainly caused by congested network links. Each flow should identify congestion situations and react accordingly by reducing its output rate. Such flows are denoted as responsive flows, as they react on congestion indications by implementing a congestion control algorithm. TCP, as the most popular transport protocol, has a built-in end-to-end rate control algorithm [1]. TCP adjusts its rate to changes in network parameters by using an *Additive Increase Multiplicative Decrease (AIMD)* [2] rate algorithm. If no packet loss is detected, TCP increases its rate constantly each round-trip time (additive increase). The sender detects a packet loss whenever it receives a negative acknowledgment or if a positive acknowledgment does not arrive during the packet's retransmission timeout interval. Upon detecting a packet loss, the sender halves its sending rate once each RTT (multiplicative decrease).

During the last decade, TCP's congestion control algorithm helped avoiding congestion collapses in the Internet. However, With multimedia streaming applications getting more popularity, the bandwidth share of UDP becomes larger. In contrast to TCP, UDP does not implement a congestion control algorithm, but rather relies on the end-applications to do this. This is mainly because implementing a congestion control algorithm is intricate, so that most multimedia streaming applications avoid implementing it, hence, showing an unresponsive behavior. Such unresponsive flows pose a threat to the long-term stability of the Internet.

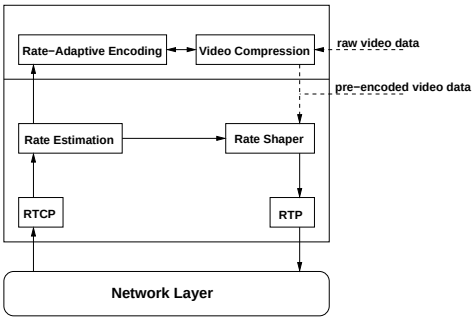

Fig. 1. Overview of the rate control components in video streaming.

The need for a congestion control algorithm, to make greedy flows (like video streaming applications) more responsive, becomes evident. Congestion control can be performed in form of rate control. Rate control can either be probe-based [1] or equation-based [3]. Probe-based rate control algorithms probe for available network bandwidth by slowly increasing their sending rate, when no congestion is detected. When congestion is detected, the rate control algorithm reduces the sending rate greatly. Equation-based rate control algorithms use a rate equation, which is proportional to the observed loss rate and round trip time, in order to calculate the available bandwidth.

After estimating the available rate, following one of the two possible algorithms, the rate control algorithm has to constraint the source output rate accordingly. Rate control for video streaming applications can be achieved in two forms: rate-adaptive encoding or rate shaping. Figure 1 gives an overview of the rate control components and their interactions.

Rate-adaptive encoding alters video compression parameters to adjust the output rate of the encoder. Given a target output bitrate, the encoder tries to achieve lowest distortion at a lower output rate. Rate-adaptive encoding can only be applied in the presence of information about the transmission channel. This is only possible for real-time video streaming applications. By contrast, rate shaping techniques operate on pre-encoded video streams to alter their original rate. This makes them suitable both for real-time and pre-encoded video streaming applications.

In the following section, we review existing approaches to comprise rate control in video streaming applications. We then introduce a distortion optimized rate control technique, that can be applied at proxy caches to adapt the rate of the video

to the available connection bandwidth of the receiver, while maximizing video quality.

II. Related Work

In the following we discuss proposed rate estimation and shaping algorithms

A. Rate Estimation

Rate estimation can be probe-based or equation-based. Using a probe-based algorithm, the sender probes the network for the available bitrate. One class of probe-based algorithms are the AIMD algorithms. Rejaie et al. [4] proposed an AIMD algorithm, where the source performs rate adaptation based on acknowledgments sent by the receivers. Rate adaptations are performed once per round trip time. When no loss is detected, the rate is increased linearly. If packet loss is detected, the sending rate is decreased multiplicatively. In RAP, the rate is adapted by changing the gap between sending two consecutive packets. Furthermore, RAP uses a ratio of short-term to long-term averages of RTT to adjust the sending rate, depending on the variations in the round trip time. Usually a window is used to determine the number of packets to send in each rate adaptation period. The window size is either increased or decreased to mimic TCP behavior.

Bansal and Balakrishnan [5] suggested class of binomial algorithms, which generalize AIMD algorithms. Binomial algorithms increase the rate inversely proportional to a power k window size, and decrease the window size proportional to a power l of the window size. Both operations are shown in the following equation.

$$\text{Increase: } w \leftarrow w + \frac{\alpha}{w^k} \tag{1}$$

$$\text{Decrease: } w \leftarrow w - \beta w^l \tag{2}$$

To obtain TCP behavior, we set $\alpha = 1, k = 0, l = 0, \beta = 0.5$.

Unlike the probe-based algorithms, the model-based rate estimation operates on the basis of a rate estimation equation. Typically, the equation used is an estimation of the throughput, which a TCP flow will achieve under the same network conditions. The TCP throughput can be estimated following equation 3 [6].

$$R_{TCP} = \frac{1.22\overline{ps}}{\overline{RTT}\sqrt{p}} \tag{3}$$

Here, p represents an estimation of the packet loss rate, $\overline{RTT}$ is a smoothed exponentially moving average of the round trip time, and $\overline{ps}$ is the average packet size. Using the feedback of the receiver, the sender can estimate its allowable sending rate. In this way, the video streaming application shows a responsive behavior for congestion indications and competes fairly with TCP flows. Flows that implement a model-based rate control algorithm are also denoted as TCP-friendly flows. The TCP-friendly Rate Control Protocol (TFRC) [7], uses a more sophisticated equation to estimate the TCP rate more accurately.

B. Rate Shaping

Rate shaping is applied to compressed video streams to adjust their rate to the available network bitrate. After the target bitrate has been determined, a rate shaping algorithm is applied to determine which portions of the video stream should be re-encoded or dropped. Different from rate-adaptive encoding techniques, rate shaping does not require interaction with the encoder, and can hence be applied to real-time as well as pre-encoded video streaming. Furthermore, rate shaping can be applied by the sender, receiver, or by proxy caches located at intermediate nodes within the network. Rate shaping can be realized using compression techniques [8] or transport techniques [9]–[11].

Layer dropping or adding is an example of transport-based rate shaping, where the receiver (or sender) decides about which layers should be received (resp. transmitted) according to the target bitrate. Another method to reduce the rate of compressed video streams uses frame dropping. The rate control algorithm is able to identify the boundaries of compressed video frames and drops a frame, whenever the target rate is exceeded.

An example of compression-based rate shaping, is presented in [8]. The technique consists of dropping DCT coefficients of high frequencies to achieve the target rate. Another technique would re-encode the video stream with higher quantizer steps or in a different format to achieve the target bitrate. However, compression-based rate shaping techniques are usually computationally intensive and are usually dismissed for delay-constrained video streaming.

III. Packet Distortion Estimation

Accurate distortion estimation for each single video packet can be exploited to develop optimal prioritized transmission schemes. In [12], [13], a statistical model for analyzing error propagation for predictive encoded video is presented. The variance of the introduced error signal is considered to be constant for the same video sequence. Two points of the packet loss rate-distortion curve are then sufficient to determine the expected distortion. However, this model cannot be used to determine the distortion value of each single video packet. Zhang et al. [14], [15] suggest the pixel-based distortion estimation of the end-to-end distortion. In their approach, the expected distortion is calculated for a group of packets by simulating a subset of the possible loss events under a given packet loss rate p of the channel. The selected subset of packet loss events for the group of packets is then simulated to get exact measures of the resulting distortion. The distortion estimation for the rest possible loss events is then derived from the measured values through a firstorder taylor approximation. This approach involves eminent amount of calculations to simulate several packet loss events for each group of packets of the compressed video sequence. In [16], an alternative distortion estimation algorithm is proposed by the same authors. The ROPE algorithm is a recursive estimation algorithm, which estimates the distortion for each single pixel by tracking its prediction chain and all possible error concealment decisions. This approach has two major

drawbacks: tracking all the possible decoder outcomes for each single packet is very hard and requires intensive processing, and on the other hand the distortion measure does not reflect the importance of a single packet, since the distortion is not followed up to the next INTRA refresh rate. It also requires the availability of the packet loss rate during estimation, which limits its applicability to the real-time video delivery applications and by consequence should work under very tight processing time limits. He et al. [17] propose a simplified macroblock-based distortion estimation, where the distortion of a single macroblock is supposed to be a constant fraction of the distortion of the whole video image. However, experiments have shown that different packets of the same frame had different distortion values, since some regions of the video frame may have a large motion amount, whereas other regions show little changes to the previous video frame. Hence, it is important to estimate the distortion of each single packet taking into account effects of error propagation to succeeding frames.

A. Distortion estimation for base-layer video packets

In order to estimate the distortion caused by a packet loss, we have to measure the difference in quality between the encoded and the concealed region of the video data carried within that packet. If the packet is received correctly and in time, the distortion measured will reflect the encoding distortion (i.e. the quantization error). If, however, the packet is lost, the decoder will conceal the lost region of the picture using older data. We assume that the decoder implements a simple error concealment technique, where a lost region is replaced by data from the same region in the previous frame. As a measure for the video quality, we consider using the widely accepted peak signal-to-noise ratio (**PSNR**). The distortion of packet P at frame t is then expressed as in equation 6.

$$\mathcal{D}_e(P,t) = -10\log(\frac{\sum_{y=0}^{height}\sum_{x=0}^{width}\left(f_e^t(x,y)-f_o^t(x,y)\right)^2}{width\times height\times 255^2}) \tag{4}$$

$$\mathcal{D}_c(P,t) = -10\log\left(\frac{\sum_{y=0}^{height}\sum_{x=0}^{width}\left(f_c^t(x,y)-f_o^t(x,y)\right)^2}{width\times height\times 255^2}\right) \tag{5}$$

$$\begin{aligned}\mathcal{D}(P,t) &= \mathcal{D}_e(P,t)-\mathcal{D}_c(P,t)\\ &= 10\log\left(\frac{\sum_{y=0}^{height}\sum_{x=0}^{width}(f_c^t(x,y)-f_o^t(x,y))^2)}{\sum_{y=0}^{height}\sum_{x=0}^{width}(f_e^t(x,y)-f_o^t(x,y))^2}\right)\\ &= 10\log\left(1+\frac{\sum_{(x,y)\in P}\left((f_c^t(x,y)-f_o^t(x,y))^2-(f_e^t(x,y)-f_o^t(x,y))^2\right)}{\sum_{y=0}^{height}\sum_{x=0}^{width}(f_e^t(x,y)-f_o^t(x,y))^2}\right)\end{aligned} \tag{6}$$

where $f_o^t(x,y)$, $f_e^t(x,y)$ and $f_c^t(x,y)$ are the luminance values of pixel at position (x,y) at frame t of the original, error-free, and concealed video sequences respectively. Furthermore, because of the motion compensation technique, deployed in all relevant video compression algorithms, the degradation in video quality may last up to the next INTRA coded frame. This effect is denoted by error propagation. We then summarize the distortion of one packet as shown in equation 7,

$$\mathcal{D}(P) = \sum_{i=0}^{n}\mathcal{D}(P,t+i) \tag{7}$$

n represents the number of frames until the next INTRA frame. We found out that $\mathcal{D}(P,t+i), i>0$ can be approximated as in equation 8,

$$\mathcal{D}(P,t+i)=10log\left(1+\frac{\alpha^i\sum_{(x,y)\in P}\left((f_c^t(x,y)-f_o^t(x,y))^2-(f_e^t(x,y)-f_o^t(x,y))^2\right)}{\sum_{y=0}^{height}\sum_{x=0}^{width}(f_e^{t+i}(x,y)-f_o^{t+i}(x,y))^2}\right) \tag{8}$$

where α expresses the attenuation of the error signal over time, which may be due to the deployment of non-integer motion compensation or some filtering techniques. During the preprocessing of the video sequence, the values of $\sum_{(x,y)\in P}\left((f_c^t(x,y)-f_o^t(x,y))^2-(f_e^t(x,y)-f_o^t(x,y))^2\right)$ and $\sum_{y=0}^{height}\sum_{x=0}^{width}(f_e^{t+i}(x,y)-f_o^{t+i}(x,y))^2$ are calculated for each of the video packets and video frames. This information is transmitted as out-of-band data to the receiver and to the proxy cache.

B. Distortion Estimation for FGS video packets

In the following, we present an algorithm to estimate the distortion for enhancement layer video packets in the example of FGS layered encoding. FGS relies on DCT encoding of the residue image to produce the enhancement layer. The residue image is the image built from the difference between an original video image and its processed image (produced after encoding and decoding). The FGS enhancement layer encoder first transforms the residue image using a DCT transform. It then performs an optional bitplane shift to change the appearance order of the macroblocks depending on their visual significance. Thereafter, the encoder makes a search for the maximal amount of bitplanes for each color component of a macroblock. At last the bitplanes are variable-length encoded.

The distortion measured in the case where only base layer video packets is received and in the case where all FGS video packets are received, are given for frame t in equations 9 and 10 respectively.

$$\xi_{\text{BL}}(t) = \sum_{y=0}^{\text{height}}\sum_{x=0}^{\text{width}}\left(f_p^t(x,y)-f_o^t(x,y)\right)^2 \tag{9}$$

$$\xi_{\text{FGS}}(t) = \sum_{y=0}^{\text{height}}\sum_{x=0}^{\text{width}}\left(f_f^t(x,y)-f_o^t(x,y)\right)^2 \tag{10}$$

$f_f^t(x,y)$ denotes the value measured at pixel $(x,y)^t$ after base and FGS layer are completely decoded. The enhancement in video quality, that can be achieved by receiving and decoding the whole FGS bitstream, is then expressed in the following equation.

$$\begin{aligned}\Delta\xi(t) &= \xi_{\text{FGS}}(t)-\xi_{\text{BL}}(t)\\ &= \sum_{y=0}^{\text{height}}\sum_{x=0}^{\text{width}}\left(f_p^t(x,y)-f_o^t(x,y)\right)^2\end{aligned}$$

$$- \sum_{y=0}^{\text{height}} \sum_{x=0}^{\text{width}} \left(f_f^t(x,y) - f_o^t(x,y)\right)^2 \quad (11)$$

Similar to the processing done for the base layer, we use the macroblock coordinates to determine the area covered by an FGS video packet and calculate the video quality gain for this specific area. In the following, we denote a packet which belongs to a bitplane i of a frame t by $P_{m,i}^t$, where m is the index of that packet. For an area A covered by video packet $P_{m,i}^t$ of video frame t, the gain in video quality in the area, after full FGS decoding, can be measured as in the following equation.

$$\xi\left(A(P_{m_i}^t)\right) = \sum_{(x,y)\in A(P_{m,i}^t)} \left(\left(f_p^t(x,y) - f_o^t(x,y)\right)^2 - \left(f_f^t(x,y) - f_o^t(x,y)\right)^2\right) \quad (12)$$

However, the same area of the video frame is covered several times, namely at each single bitplane. To estimate the distortion of a single packet, we have to partition the quality gain measured for the covered area among the packets that cover this area at all biplanes. We introduce a distortion partition model, which partitions the distortion value of an area among all bitplanes, depending on their significance and size. We recall that bitplanes are bits taken at different significance levels of the DCT coefficients of the residue image. This means that a bit in bitplane 0, which is the most significant bitplane (MSB), is two times as significant as a bit taken from bitplane 1. In general, the bits at bitplane bp_i are twice as important as bits at bitplane bp_{i+1}. On the other hand, the amount of data present at a bitplane of higher significance is generally smaller than that of lower significant bitplane. This is mainly because more detail of the video image is present in low significance bitplanes. We relate the amount of detail present in a bitplane to the size of the encoded data of that bitplane. By consequence, a video packet of larger size has more detail of the video image than another video packet of smaller size. A further specificity of FGS layered coding, is that the video stream is designed to be truncated at any position. However, the decoder will not be able to recover from a packet loss and continue decoding the FGS data of the same video frame. By consequence, if a packet is lost, then all the following packets of the same frame are useless and will be discarded at the decoder. We reflect this dependence in our model by assigning each packet a distortion value equal to the sum of the quality gain of that packet and all dependent packets of the same FGS frame. For a packet $P_{n,j}^t$, which depends from a packet $P_{m,i}^t$, we express this dependency by a the following notation: $P_{n,j}^t \subset P_{m,i}^t$.

We now are able to estimate the distortion value of a single packet of the FGS enhancement layer. Given the packet $P_{m,i}^t$, at frame t and that carries data of bitplane i, the distortion value of this packet is estimated by the following equation,

$$\xi(P_{m,i}^t) = \xi\left(A(P_{m,i}^t)\right) \times \frac{2^{\text{maxbps-1-i}} size(bp_i)}{\sum_{l=0}^{maxbps-1} 2^{maxbps-1-l} size(bp_l)} \quad (13)$$

$$\Delta D(P_{m,i}^t) = 10 \times \log\left(\frac{\xi(P_{m,i}^t) + \sum_{P_{n,j}^t \subset P_{m,i}^t} \xi(P_{n,j}^t)}{\xi_{FGS}(t)}\right) \quad (14)$$

where maxbps is the number of bitplanes encoded, $size(bp_i)$ is the size of bitplane i, $i = 0$ is the most significant bitplane, and $D(A(P_{m,i}^t))$ is the area covered by data of FGS packet $P_{m,i}^t$. All the information needed to estimate the distortion can be read from the encoded FGS bitstream. To measure the distortion of the area A of the frame t, we need the original video image, the decoded BL video image and BL+FGS layer video image.

The presented model does not take into account the case, where the video data of the base layer is lost. In that case, we assume that the encoder will not make use of the FGS enhancement layer data, since this may cause further damage to the concealed video image. A clear priority between packets of the base layer and enhancement layer exists, which should be taken into account when implementing priority schemes for video streaming.

IV. Rate Estimation for Video Delivery Services

In this section, we introduce a TCP-friendly rate estimation algorithm, Additive Increase Multiplicative Decrease with Congestion Avoidance (*AIMD-CA*). AIMD-CA does not assume synchronous feedback from the receiver. In contrast to TCP, RTP does not support the acknowledgment of each single packet received. Instead, RTCP is used to report loss ratio and delay jitter information to the session participants periodically. Hence, only coarse grain rate control can be performed. Our rate estimation algorithm tries to avoid congestion by linearly reducing the sending rate, when the delay jitter is growing. A growing delay jitter is assumed to be a congestion notification and hence congestion avoidance is applied to reduce the rate. In case packet losses are detected, the algorithm reduces the sending rate multiplicatively by a factor of $\frac{1}{2}$. A slow-start is avoided, to reduce the oscillations of the output rate. If further losses during the same round trip time are detected, the rate remains unchanged, since we assume that the rate reduction did not show its effects yet. In case no packet loss is detected, the rate is increased additively by one packet each round trip time. The algorithm is implemented using a congestion window, which gives for each round trip time, the amount of packets to send. The algorithm is summarized as follows.

- if packet losses are observed, set

$$cwnd \leftarrow \frac{cwnd}{2} \quad (15)$$

  under the assumption that

$$t_{\text{current}} - t_{\text{last change}} > RTT \quad (16)$$

  $t_{\text{last change}}$ is updated accordingly.
- if $RTT_{cur} - RTT_{last} > (1+\gamma)\Delta_{last}$ and $RTT_{cur} > RTT_{last}$, then we decrease the window size as follows

$$cwnd \leftarrow cwnd - 1 \quad (17)$$

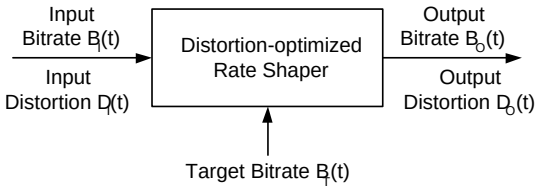

Fig. 2. Distortion-optimized rate shaping (DORS).

$\gamma \geq 0$ is used to tune the sensitivity of the algorithm to changes in the delay jitter. we update the parameters as follows

$$\Delta_{last} \leftarrow RTT_{cur} - RTT_{last} \text{ and } RTT_{last} \leftarrow RTT_{cur}. \quad (18)$$

- else, the window size is increased linearly as follows $cwnd \leftarrow cwnd + 1$.
- the output rate is then set to be

$$r = max\{cwnd\frac{\overline{ps}}{\overline{RTT}}, \frac{MTU}{\overline{RTT}}\} \quad (19)$$

where $\overline{ps}$ is the average packet size, $\overline{RTT}$ is the smoothed round trip time, and MTU is the path MTU. This should ensure that at least one packet per round trip time is sent.

V. Distortion-Optimized Rate Shaping

In this section, we introduce a new rate shaping algorithm, the *distortion-optimized rate shaping (DORS)* algorithm. The DORS algorithm performs rate shaping of a compressed video stream. Given a target bitrate, DORS determines which video packets should be sent, in order to maximize the perceptual quality while not exceeding the specified rate. The distortion-optimized rate shaping problem is described in figure 2.

For a given target bitstream $B_T(t)$, a total input distortion $D_I(t)$, and an input rate $B_I(t)$ at an time t, our goal is to minimize the introduced distortion $D_O(t) - D_I(t)$ at an output rate $B_O(t) \leq B_T(t)$.

DORS takes into account the existing error control techniques, like FEC and retransmissions. An algorithm to determine the optimal rate allocation among the data of the base layer, the enhancement layer video, and the FEC redundancy packets is presented. Video packets which are retransmitted are treated as usual base layer data packets, whereas no retransmission is performed for enhancement layer video packets or FEC redundancy packets.

The DORS algorithm uses a leaky bucket to assure that the output rate is conformant with the specified target rate. Figure 3 depicts the leaky bucket mechanism. A leaky bucket functions in a similar way to a bucket with a hole downside, which pours fluid at a rate r (if fluid exists inside the bucket). A leaky bucket is inserted at the output of the flow, which is to be controlled. The data packets are inserted into the bucket, which in turn will forward them to the channel transmitter at the specified target rate. The size of the bucket represents the maximum amount of fluid that can be inserted into the bucket without loss. In other words, the size of the bucket reflects the acceptable burstiness of the traffic. If the incoming data traffic brings the leaky bucket to overflow, then the excessive

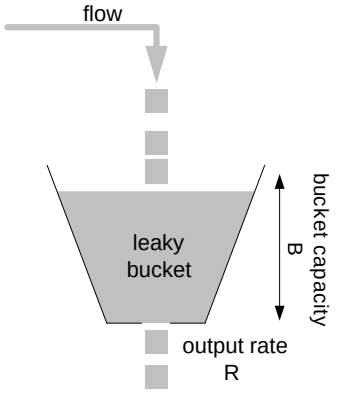

Fig. 3. Leaky bucket deployed for rate control.

packets are dropped. Excessive packets are also denoted as non-conformant packets, since they do conform to the target rate curve of the flow. The target rate curve of the flow is specified by the target rate and the permitted burst size. A bucket underflow takes place when no data packets are available for transmission in the leaky bucket.

On the other hand, if the leaky bucket capacity is exceeded, the non-conformant packets are discarded. In general, the non-conformant packets are the packets that were lastly received. In this work, we propose different packet discarding mechanisms and compare their performance. In all cases, the leaky bucket has to keep track of its current size and to be able to read the size of the incoming packets. A simple comparison operation ($currentsize + packetsize <= maximalcapacity$) is then performed to check if an overflow will occur.

1) Simple Tail Drop Rate Shaping: In the Simple Tail Drop algorithm, the non-conformant packets, which have caused the overflow of the leaky bucket, are discarded. Upon receiving a new packet, the leaky bucket checks if the new packet will lead to an overflow, if the packet is added to the bucket. If this is the case, the packet is discarded. If the bucket capacity is not exceeded by adding the new packet, the packet is inserted at the tail of the leaky bucket. This mechanism is very simple and can be implemented easily.

2) Distortion-optimized Rate Shaping (DORS): The DORS algorithm aims at minimizing the overall distortion in the perceptual quality. If an overflow is identified, the leaky bucket minimizes the overall expected distortion by discarding the appropriate packets. This is achieved by solving the optimization problem described in the following. Let the number of packets currently present in the leaky bucket be n. We enumerate the packets by an index from 1 to n. A packet i has a size s_i and a distortion value $d_i = \mathcal{D}(P)$ (calculated according to the distortion models presented previously). The optimization problem can then be described by equation 20.

$$\arg\min_{x=\{x_1,..,x_n\}} \sum_{i=1}^{n} ((1 - x_i) \times d_i)$$

subject to

$$\sum_{i=1}^{n} (x_i \times s_i) \leq S_{target} \tag{20}$$

where

$$x_i = \begin{cases} 1 & \text{packet } i \text{ is kept in the leaky bucket} \\ 0 & \text{packet } i \text{ is discarded} \end{cases} \tag{21}$$

In other words, we have to minimize the overall distortion of the discarded packets, while keeping the size of the kept packets under the specified leaky bucket target size S_{target}. The solution of this problem is a vector $\overset{*}{x} = \{\overset{*}{x_1}, .., \overset{*}{x_n}\}$, which indicates for each packet i if it is to be kept in the leaky bucket or discarded. We choose S_{target} to be smaller than the maximal leaky bucket capacity, so that the optimization algorithm is not ran each time a new packet is received.

An efficient relaxation, that might be applied to get an approximation for the solution is the so called linear programming relaxation to reach the Dantzig's bound [18]. The approach divides the set of packets into three subsets. The first subset contains items, which are kept, i.e. $x_i = 1$. The second subset contains a single item, which is fragmented. Only one fragment of the item is included. The last subset contains all items for which $x_i = 0$. We extend this algorithm as follows,

1) for each packet i, calculate $r_i = \frac{d_i}{s_i}$.
2) sort the packets according to their r_i values in a descendant manner.
3) start from the packet with the highest r_j and set $x_j = 1$. set $C = s_j$.
4) check for the following packet i if $C + s_i \leq S_{target}$. If this is the case set $x_i = 1$ and $C \leftarrow C + s_i$. If not, then mark this packet i as the critical item.
5) optionally check for the following packets if they fit into the left space.

This algorithm can be implemented recursively, in order to limit the search time. Each time a new packet is added or a packet is removed from the set of packets, the critical packet is determined again. If a packet has to be dropped, the search is performed only for the set of packets, which have a lower r_i than that of the critical packet.

3) Layer-based Distortion-optimized Rate Shaping (LDORS): The LDORS algorithm is an extension to the DORS mechanism. In LDORS, the packets are first classified according to the corresponding video layer of the video packet. A higher priority is given to the base layer video packets. In the case of an overflow, the leaky bucket tries to discard packets from the enhancement layer first. This is performed using the DORS mechanism over all enhancement layer packets, with modified target size and current size parameters. If the overflow still exists after treating enhancement layer packets, the DORS mechanism is again applied to the base layer packets. The algorithm is described as follows.

1) set $S_{drop} = S_{cur} - S_{target}$, which is the number of bytes that need to be dropped.
2) set S_{enh} to the size of all enhancement layer packets.
3) if $S_{drop} < S_{enh}$ then we perform the DORS on the set of enhancement layer packets, where the target size for the optimization is set to $S_{enh} - S_{drop}$.
4) if $S_{drop} = S_{enh}$ then we drop all enhancement layer packets.
5) if $S_{drop} > S_{enh}$ then we first drop all enhancement layer packets. We set S_{base} to the total size of the base layer packets. Then, we perform the DORS on the set of all base layer packets. The target size for the optimization is set to $S_{base} - (S_{drop} - S_{enh}) = S_{target}$.

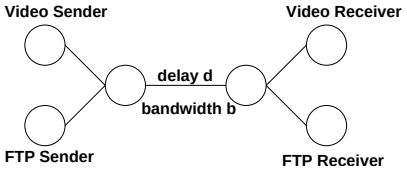

Fig. 4. Topology used for simulations to compare the different rate estimation algorithms.

VI. Evaluation

In this section we evaluate the proposed rate estimation and rate shaping algorithms by comparing them to other algorithms. We start by comparing our proposed rate estimation algorithm AIMD-CA to the class of binomial algorithms and equation-based rate estimation algorithm. We rely solely on RTCP feedback to estimate the channel conditions. RTCP receiver reports are used to estimate the round trip time and the packet loss rate. For our experiments, we used the ns-2 simulator and the network topology shown in figure 4.

We first measure the fairness ratio, which is the ratio of video to TCP throughput $\left(\frac{\text{Video Throughput}}{\text{TCP Throughput}}\right)$, of the video connection using different rate estimation algorithms. We varied the delay time d of the bottleneck link while keeping the bandwidth constant at $400kbps$. The video server is sending an FGS layered video sequence with a total rate of about $400kbps$. To adjust the rate, the video server drops some of the video packets using our rate shaping algorithm. The results are depicted in figure 5. We observe that the equation-based rate estimation algorithm is too aggressive in adjusting the sending rate. TCP connections get a much higher share of the bottleneck bandwidth than the video application. The reason for this is the long rate adaptation period, which is dependent on RTCP feedback. Binomial algorithms are less friendly than our AIMD-CA algorithm, and achieve a throughput of about $4 \sim 8$ as that of the TCP connection. The AIMD-CA algorithm is more friendly to TCP and achieves a $2 \sim 4$ higher throughput than that of TCP. We also observe the increase of the throughput of the video connection for all but the equation-based rate estimation algorithm with increasing bottleneck delay.

Figure 6 depicts the confidence interval for the AIMD-CA algorithm at different bottleneck delays. The variation in the ratio of video to TCP throughput increases with higher bottleneck delays. However, the variations are of small range, which shows that the AIMD-CA algorithm is stable.

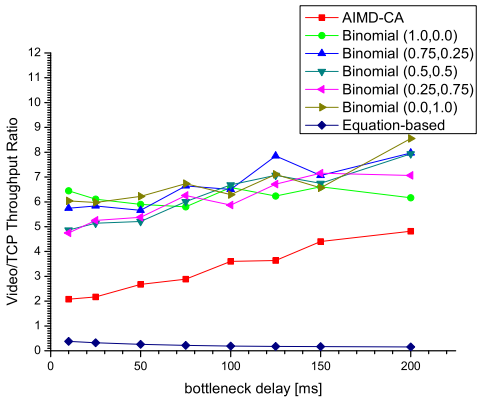

Fig. 5. Ratio of video to TCP throughput vs. bottleneck delay.

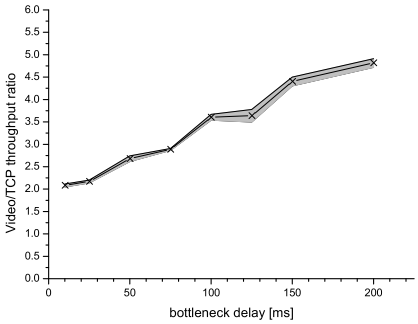

Fig. 6. Ratio of video to TCP throughput vs. bottleneck delay with confi dence interval for the AIMD-CA algorithm.

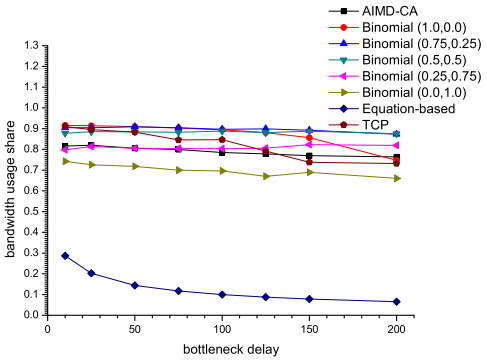

Fig. 7. Usage share of bottleneck bandwidth vs. bottleneck delay.

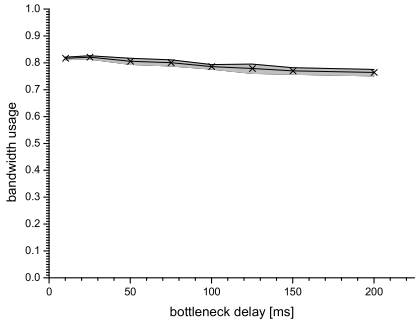

Fig. 8. Usage share of bottleneck bandwidth vs. bottleneck delay with confi dence interval for the AIMD-CA algorithm.

In a second simulation, we vary the bandwidth of the link and measure the bandwidth usage, which is the $\left(\frac{\text{connection throughput}}{\text{average available bandwidth}}\right)$, of the video connection using the different rate estimation algorithms. In this simulation, a single connection is used. We start with a bottleneck bandwidth of $200kbps$, and set the bottleneck bandwidth to $400kbps$ and then to $100kbps$. We also simulated the behavior of a TCP connection for comparison reasons. Figure 7 shows the measured bandwidth usage for the different rate estimation algorithms. The equation-based rate estimation algorithm was not able to adapt to the bandwidth changes of the bottleneck. The remaining algorithms achieved a bandwidth usage of $0.7 \sim 0.9$ of the available bottleneck bandwidth. At lower bottleneck delay, TCP achieved higher bandwidth usage. AIMD-CA behaved almost independently of the bottleneck delay.

Figure 8 depicts the confidence interval for the bandwidth usage of the AIMD-CA algorithm for different bottleneck delays.

We now compare the different rate shaping policies introduced previously. We also consider the case where no rate shaping is performed. In the first simulation, we varied the bottleneck delay and started a concurring TCP connection. We then measured the video quality for the following algorithms,

- no rate control is performed.
- simple tail dropping is performed, where the non-conformant packets are simply dropped.
- Layer-based dropping, where tail dropping is first performed on enhancement layer packets, and then on base layer packets if necessary.
- DORS
- LDORS

The leaky bucket capacity was set to 10000 Bytes. The same layered video sequence was used for the simulations. In figure 9, we depict the measured video quality for the different scenarios. The case where no rate control is performed achieved the highest video quality, but was outperformed by the DORS algorithm for low bottleneck delays. In the case where no rate control is performed, the available bandwidth is fully used by the video connection at the cost of the TCP connection. However, DORS may still achieve higher video quality, while being TCP-friendly.

The results have also shown, that distortion-based dropping is more efficient than the layer-based dropping. DORS outperformed both the layer-based and the LDORS algorithms. The reason for this, is that some enhancement layer packets are more important than base layer packets. This fact is ignored by the layer-based and the LDORS algorithms. Although tail dropping is the simplest algorithm, it achieved lowest video quality. Figure 10 shows the 95% confidence interval of the average video quality for the DORS algorithm.

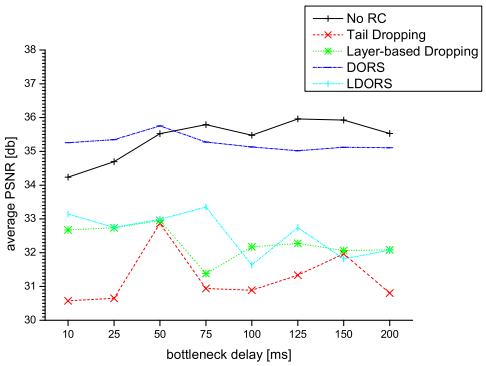

Fig. 9. Video quality vs. bottleneck delay.

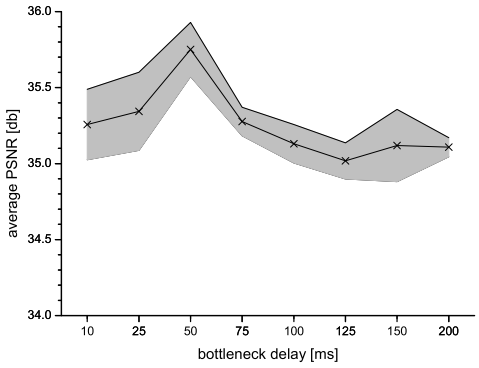

Fig. 10. Video quality vs. bottleneck delay with confidence interval for the DORS algorithm.

Figure 11 depicts the video quality for the different dropping algorithms for the bottleneck delay of 50 ms. For the case where no rate control is performed, fluctuation of the achieved video quality can be observed. DORS achieved a more or less stable video quality and outperformed all other dropping algorithms.

We varied the allowable burst size of the leaky bucket and measured the video quality for the DORS algorithm. The results are shown in figure 12. The video quality measured increased with the leaky bucket capacity. Depending on the playout delay configured at the receiver, higher capacity of the leaky bucket, which introduces an increasing delay, leads by consequence to late arrivals for the video packets. The bottleneck delay was set to $25ms$ and the playout delay to $1s$.

VII. Summary

In this chapter, we discussed the different types of rate control for video streaming over best-effort networks. We proposed a rate estimation algorithm AIMD-CA, which achieved TCP friendliness under coarse grain rate adaptation, based on RTCP feedback. The results have shown the good performance of AIMD-CA as compared to other rate estimation algorithms. Based on this rate estimation model, we introduced the DORS rate shaping algorithm, which determined the packets to send

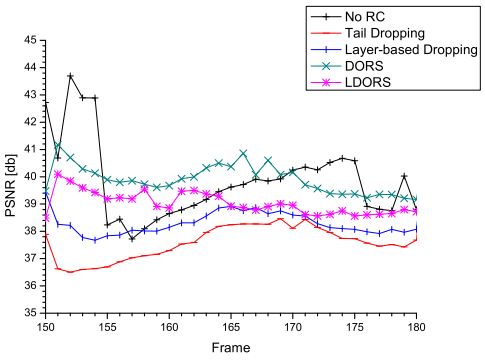

Fig. 11. Video quality for the different algorithms at bottleneck delay 50ms.

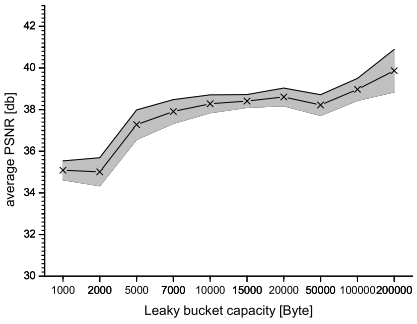

Fig. 12. Video quality vs. maximum burst size with confidence interval for the DORS algorithm.

under a given rate constraint, in order to minimize the overall distortion. The DORS algorithm is based on a leaky-bucket, which shapes the video traffic. We deployed the rate shaping algorithm in a proxy cache close to the receiver, so that the rate of the video stream is adapted to the channel rate of the receiver. Significant improvements were achieved in the video quality as compared to other rate shaping algorithms.

References

[1] V. Jacobson, "Congestion avoidance and control," in *Symposium proceedings on Communications architectures and protocols*, Stanford, California, August 1988, pp. 314–329.

[2] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. Vol. 17, no. No. 1, pp. 1–14, June 1989.

[3] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," in *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Basking Ridge, NJ, June 1999, pp. 73–88.

[4] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end congestion control mechanism for realtime streams in the internet," in *Proceedings of the Infocom*, New York, NY, March 1999, pp. 1337–1345.

[5] D. Bansal and H. Balakrishnan, "TCP-friendly congestion control for real-time streaming applications," Tech. Rep., MIT Technical Report, MIT-LCS-TR-806, 2000.

[6] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, vol. Vol. 7, pp. pages 458–472, August 1999.

[7] Mark Handley, Sally Floyd, J. Padhye, and Joerg Widmer, "TCP friendly rate control (TFRC): Protocol specification," Tech. Rep., Internet Engineering Task Force, January 2003.

[8] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in *Proceedings of the IEEE Internation Workshop on Networks and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, April 1995, pp. 95–106.

[9] M. Hemy, U. Hengartner, P. Steenkiste, and T. Gross, "MPEG system streams in best-effort networks," in *Proceedings of the IEEE Packet Video'99*, New York, NY, April 1999, pp. 215–221.

[10] K. Sripanidkulchai and T. Chen, "Network-adaptive video coding and transmission," in *Proceedings of the Visual Communications and Image Processing (VCIP'99)*, San Jose, CA, January 1999, pp. 166–177.

[11] Z.-L. Zhang, S. Nelakuditi, R. Aggarwa, and R. P. Tsang, "Effi cient server selective frame discard algorithms for stored video delivery over resource constrained networks," in *Proceedings of the Infocom'99*, New York, NY, March 1999, pp. 472–479.

[12] Klaus Stuhlmüller, Niko Färber, Michael Link, and Bernd Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on selected areas in Communications*, vol. Vol. 18, no. No. 6, pp. pages 1012–1032, June 2000.

[13] Niko Färber, Klaus Stuhlmüller, and Bernd Girod, "Analysis of error propagation in hybrid video coding with application to error resilience," in *IEEE International Conference on Image Processing*, Kobe, Japan, October 1999, pp. 550–554.

[14] R. Zhang, S. L. Regunathan, and K. Rose, "End-to-end distortion estimation for RD-based robust delivery of pre-compressed video," in *Proceedings of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, 2001, vol. 1, pp. 210–214.

[15] R. Zhang, S. Regunathan, and K. Rose, "Optimized video streaming over lossy networks with real-time estimation of end-to-end distortion," in *Proceedings of the International Conference on Multimedia and Expo*, Lausanne, Switzerland, August 2002, pp. 861–864.

[16] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE Journal on Selected Areas in Communications*, vol. Vol. 18, pp. pages 966–976, June 2000.

[17] Z. He, J. Cai, and C. W. Chen, "Analytic end-to-end rate distortion modeling and control for packet video over wireless network," in *Proceedings of the International Packetvideo Workshop*, Pittsburgh, PA, April 2002, pp. 115–127.

[18] G. B. Dantzig, "Discrete variable extremum problems," *Operations Research*, vol. 5, pp. 266–277, 1957.