Hindawi Publishing Corporation
Advances in Multimedia
Volume 2008, Article ID 628970, 13 pages
doi:10.1155/2008/628970

Research Article

Rate-Distortion Optimized Frame Dropping for Multiuser Streaming and Conversational Videos

Wei Tu,[1] Jacob Chakareski,[2] and Eckehard Steinbach[1]

[1] *Media Technology Group, Institute of Communication Networks, Munich University of Technology, 80333 Munich, Germany*
[2] *Vidyo Inc., Hackensack, NJ 07601, USA*

Correspondence should be addressed to Wei Tu, wei.tu@tum.de

Received 11 May 2007; Accepted 14 September 2007

Recommended by Zhu Li

We consider rate-distortion optimized strategies for dropping frames from multiple conversational and streaming videos sharing limited network node resources. The dropping strategies are based on side information that is extracted during encoding and is sent along the regular bitstream. The additional transmission overhead and the computational complexity of the proposed frame dropping schemes are analyzed. Our experimental results show that a significant improvement in end-to-end performance is achieved compared to priority-based random early dropping.

1. INTRODUCTION

In today's Internet, video packets are typically transmitted using best-effort service. The packet forwarding service at network nodes is significantly degraded if the network is congested. In this paper, we consider the scenario where K streaming videos and N conversational videos pass through a network node (e.g., a multimedia gateway) with limited forwarding resources, as illustrated in Figure 1. The packets can be temporarily cached in the node's buffer, but if the overload persists, the buffer will overflow and some packets will be lost. Our goal is to improve the overall quality of the streams for the given forwarding resource R_{out} of the node.

For video applications, transcoding [1–3] or pruning of the video stream can be performed to adapt the source rate to the available transmission rate. Transcoding is computationally expensive and not suitable for a node that has to rapidly forward packets of many different users. Furthermore, video source pruning by random frame dropping may have a dramatic influence on the reconstructed video quality. In [4–6], static priority labels for I-, P-, and B-frames are used to perform priority-based random dropping (PRD) for streaming video. In particular, video frames are dropped according to their priority labels. Random selection is performed among frames with the same priority label. Priority-based random early dropping (PRED) [7] improves PRD by early dropping of lower priority frames at certain predefined buffer fullness levels. Nonetheless, static priority labels cannot accurately describe the importance of video frames. For example, the first P-frame in a group of pictures (GOP) is in most cases much more important than the last P-frame, although they belong to the same priority class. In [8, 9], the decodability of the video frames is used to make dropping decisions.

Rate-distortion (RD) optimization has been widely employed to deal with the varying importance of video frames. In [10], for instance, it is used to achieve RD-optimized frame scheduling for a single video stream. RD-optimization for bit allocation between source coding and channel coding is used, for example, in [11]. RD-optimization is also the state-of-the-art for coding mode selection in video compression [12, 13]. An RD-based robust delivery scheme is proposed in [14, 15]. However, all these works focus on the video encoding at the sender to choose the best encoding and sending strategy according to the constrained transmission rate and the expected packet loss rate. When RD-optimization is done at intermediate nodes in the network, side information has to be transmitted along with the video stream [16–18]. In particular, [16] considers RD-optimization in a broadcast networking scenario, and performs optimization only for a single stream, while [17] employs RD side information for transcoding at network nodes. Only streaming video is considered in [18], while in this paper we also examine the

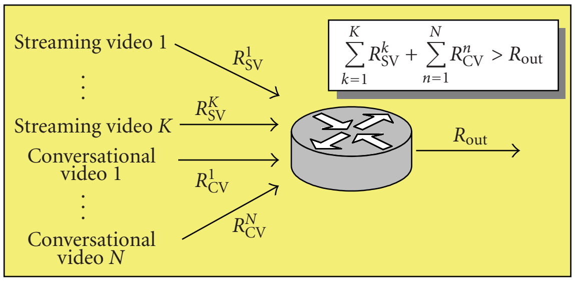


Figure 1: A network node with K incoming streaming videos and N conversational videos sharing the same outgoing link.

additional case of frame dropping for both streaming and conversational videos, simultaneously.

In the present paper, we propose RD-optimized video frame dropping strategies for streaming or conversational video on overloaded network nodes. For rate shaping streaming videos, we use a *distortion matrix* and a *rate vector* [19] as side information. We denote our approach as the cost function-based approach, which minimizes a Lagrangian cost function in order to find the optimum dropping pattern. The cost function employs the following quantities to determine the optimal pattern: the *rate vector* and the *distortion matrix* of all incoming streams, as well as the current fullness of the outgoing buffer. The Lagrangian multiplier in the cost function is selected as a function of the buffer fullness and is used to adjust the aggressiveness of the dropping process. The *distortion matrix* can be extracted only for GOP structured video. For conversational video with IPPP... structure, only *hint tracks* [20, 21] can be calculated and therefore, the utility-based frame dropping strategy is used for the conversational videos. Frame dropping decisions are made only when the buffer does not have enough space to hold them.

In addition, we also examine the scenario when both streaming and conversational videos are passing through a network node. In this case, we propose to use separate classification buffers combined with a scheduler for dynamic resource assignment to the two buffers, which is located between the two classification buffers and the outgoing link buffer at the node. For simplicity, in our framework we replace continuous time with the discrete frame slots of the video sequences, which means that dropping decisions will only be made at multiples of one frame duration. In case the streams have different frame rates, the dropping decision can be made synchronized to the stream with the highest frame rate. Another approach would be to collect a small number of frames from all incoming streams and then perform a dropping decision. This approach, however, would introduce additional delay.

The main contributions of this work include the following:

(1) *joint rate shaping for both streaming and conversational videos*,

(2) *extensive simulation results* which provide a comprehensive performance comparison of different frame dropping schemes as well as a reference for parameter selection,

(3) *analysis of computational and storage cost* which can be used as a reference to select different dropping schemes for a given scenario.

The rest of the paper is organized as follows. Section 2 describes the side information used for streaming video and the corresponding frame dropping strategies. Next, the side information and dropping strategy for conversational video are introduced in Section 3. An integrated RD-optimized framework for both streaming and conversational video applications is presented in Section 4. Furthermore, in Section 5 we analyze the memory requirements and the computational complexity of the techniques considered in this paper. Section 6 presents the simulation results that demonstrate the improvements achieved by our proposed RD-optimized frame dropping strategies. Conclusions are drawn in Section 7.

2. FRAME DROPPING STRATEGIES FOR STREAMING VIDEO

In this section, we first introduce the priority-based frame dropping schemes, which are used for comparison in this paper. Then, the definition and the procedure to construct the side information (*distortion matrix* and *rate vector*) for our approach are presented. Based on this side information, we propose an RD-optimized frame dropping strategy based on the current buffer fullness of the network node.

2.1. Priority-based random early dropping (PRED)

PRD makes dropping decisions based on fixed priority labels assigned to the video frames. With current conventional coding scheme, frames can be prioritized according to their frame type: I-, P-, or B-frame. When frames from multiple video streams simultaneously arrive at a network node, it is the I-frames among them that have the highest priority to be placed into the node's buffer. It may happen though that some of these I-frames cannot be placed into the buffer and therefore they are dropped even before the buffer is totally full. In this case, P-frames are tried next to be placed into the buffer, followed then by the remaining (if any) B-frames. This strategy leads to the most efficient usage of the node's buffer. However, the loss of I-frames and P-frames has a dramatic influence on the reconstruction video quality.

PRED sets thresholds for dropping according to the number of priorities available for the video streams. Here, we only have three priority levels $\{I, P, B\}$, so only two dropping thresholds are needed. But if we have different priority levels for all P-frames according to their positions in the GOP, we can set $n = N_G/(N_B + 1)$ thresholds, where N_G is the length of the GOP and N_B denotes the number of B frames between two I/P-frames. As shown in Figure 2, when the buffer fullness reaches T_1, the least important B-frames are all dropped. The last P-frame in the GOP is dropped when T_2 is reached. I-frames have the highest priority level and should not be early dropped, so all P-frames are dropped when the buffer fullness reaches the highest threshold (T_n). Early dropping of less important frames reduces the likelihood of having to drop more important frames at a later time.

2.2. Distortion matrix (DM) and rate vector (RV)

The *distortion matrix* proposed in [19] allows us to calculate the distortion caused by dropping frames in a GOP structured video stream. When calculating the reconstruction distortion, it is assumed that a simple "copy and freeze previous frame" error concealment scheme is employed by the decoder. In particular, a missing frame and all of its descendants[1] are replaced, at reconstruction, by the decoder with the temporally nearest previous frame that has been decoded. Note that this is done regardless of the presence status, at the decoder, of the descendant frames. Hence the name of the concealment scheme.

Our approach to frame dropping follows this logic. When a network node drops an arriving video frame, it subsequently drops all its dependent frames that arrive at the node afterwards. Therefore, a video frame drop pattern comprises in our case an incoming frame that is dropped at present together with its descendant frames that will be dropped afterwards.[2] The increase in reconstruction distortion affecting a video stream caused by a frame drop pattern is the sum of the individual increments in reconstruction distortion for the concealed video frames. That is because the frames that have been decoded do not contribute to the increase in reconstruction distortion. The *distortion matrix* for a GOP with $IB_1B_2P_1B_3B_4P_2B_5B_6$ structure is given in (1), where $D_{F_{\text{loss}}}^{F_{\text{rep}}}$ represents the increased distortion in MSE that is observed when replacing frame F_{loss} by F_{rep} as part of the concealment strategy. The column left to the matrix shows the replacement frame F_{rep} for every row of the matrix. For instance, $D_{B_1}^{I}$ represents the additional reconstruction distortion if the first B-frame of the GOP is lost and is therefore replaced by the I-frame of that GOP. R is a frame from the previous GOP that is used as a replacement for all the frames in the current GOP if the I-frame of the current GOP is lost. As a worst case assumption, we use the I-frame of the previous GOP as the replacement frame in this case:

$$\begin{array}{r} R: \\ I: \\ P_1: \\ P_2: \\ B_1: \\ B_3: \\ B_5: \end{array}\begin{bmatrix} D_I^R & D_{B_1}^R & D_{B_2}^R & D_{P_1}^R & D_{B_3}^R & D_{B_4}^R & D_{P_2}^R & D_{B_5}^R & D_{B_6}^R \\ / & D_{B_1}^I & D_{B_2}^I & D_{P_1}^I & D_{B_3}^I & D_{B_4}^I & D_{P_2}^I & D_{B_5}^I & D_{B_6}^I \\ / & / & / & / & D_{B_3}^{P_1} & D_{B_4}^{P_1} & D_{P_2}^{P_1} & D_{B_5}^{P_1} & D_{B_6}^{P_1} \\ / & / & / & / & / & / & / & D_{B_5}^{P2} & D_{B_6}^{P2} \\ / & / & D_{B_2}^{B_1} & / & / & / & / & / & / \\ / & / & / & / & / & D_{B_4}^{B_3} & / & / & / \\ / & / & / & / & / & / & / & / & D_{B_6}^{B_5} \end{bmatrix}. \tag{1}$$

The entries of the *rate vector* correspond to the sizes of the video frames expressed in bytes. Then, at a network node, the size of an incoming frame and the sizes of its descendant frames are summed up to determine the rate saving achieved by dropping these frames.

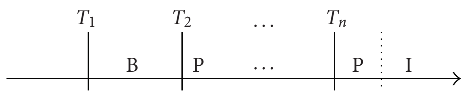


FIGURE 2: Example settings of dropping thresholds for PRED.

2.3. Cost function-based video frame dropping

In Section 2.1, we reviewed the idea of PRED and discussed the benefit obtained by "early" dropping. In this section, a cost function-based approach is proposed, which takes advantage of the RD side information to enable more flexible frame dropping decisions, while still using the buffer fullness info for early dropping.

If the buffer is empty or is lightly loaded, no frames should be dropped. However, when the buffer fills up, frames that have the least impact on the perceived quality at the receiver should be dropped first. The decision which frames to drop is jointly made in our approach for all video streams. Given the RD side information introduced in Section 2.2, the active network node can perform RD-optimized frame dropping. For this, the node checks the current buffer fullness and minimizes the Lagrangian cost function

$$J_p(n) = \sum_{k=1}^{K} \Delta D_p^k(n) - \lambda(n) \sum_{k=1}^{K} \Delta R_p^k(n) \tag{2}$$

in order to determine the optimal drop pattern. In (2), n is the current (discrete) time instant (slot), $\Delta D_p^k(n)$ is the additional distortion introduced in video k for a given drop pattern p, and $\Delta R_p^k(n)$ is the corresponding rate saving in bytes.

When the *distortion matrix* and *rate vector* described in Section 2.2 are used, a dropping decision should comply with the following rules. If the current frame that arrives at the active node is an I-frame, we can either drop this frame or send it to the outgoing link buffer. If we drop it, this means that all the following P- and B-frames in the same GOP cannot be decoded and have to be dropped also. This dropping strategy leads to a significant increase in distortion for this GOP but at the same time allows us to reduce the sending rate to 0 for this GOP. If we do not drop the I-frame at this moment, we can still decide to drop the subsequent P-frames and B-frames. This will lead to reduced distortion but also the rate saving will be smaller. We could also drop the B-frames only if we decide not to drop the P-frames. Again, the additional distortion will be reduced but also the rate saving will be even smaller.

Therefore, if the current incoming frame is an I-frame, there are in total 4 dropping choices $\{I, P, B, N\}$, where I denotes dropping the whole GOP, P stands for dropping the subsequent P- and B-frames in the GOP, while B signifies dropping the all B-frames in the current GOP only and N stands for "drop nothing." If the current frame is a P-frame,

[1] These are the frames in the encoding chain that depend on the missing frame in order to be decoded, that is, decompressed.

[2] In case they do arrive at the node.

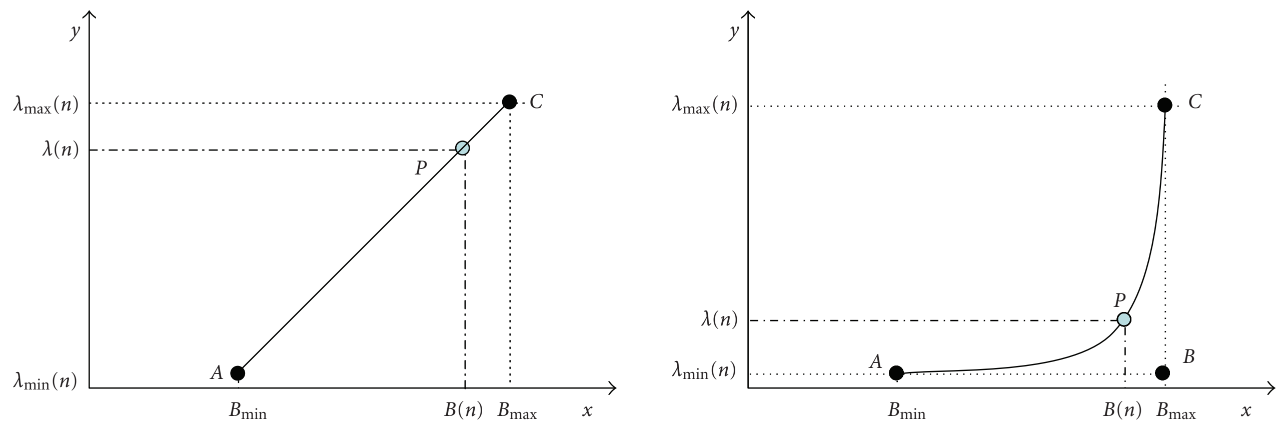


FIGURE 3: Interpolation of $\lambda(n)$ between $\lambda_{\min}(n)$ and $\lambda_{\max}(n)$ as a function of the current buffer.

the choices are reduced to $\{P, B, N\}$. If the current frame is a B-frame, the choices are also $\{B, P, N\}$, where B denotes the case of dropping all the remaining B-frames in the GOP, including the current one, and P stands for dropping the subsequent (relative to the current B-frame) P- and B-frames.

Now, if we denote the number of possible drop patterns at time n for video k as $A^k(n)$, then for K videos we obtain the dropping set $P(n)$ including $\prod_{i=1}^{K} A^i(n)$ different drop patterns. One of the drop patterns will minimize (2). This pattern represents the optimal dropping strategy at time n. In order to perform this minimization, we have to determine a reasonable value for the Lagrangian multiplier $\lambda(n)$ in (2). In this work, we determine $\lambda(n)$ as a function of the buffer fullness $B(n)$. If the buffer is empty, we certainly do not want to drop any video frames. This has to be reflected by an appropriate choice of $\lambda(n)$. On the other hand, if the buffer is full, $\lambda(n)$ should be selected such that all incoming frames are dropped as queueing them in the outlink buffer would fail anyway. In order to determine appropriate values for $\lambda(n)$ for any buffer level, we define a minimum buffer fullness $B_{\min}$, below which no dropping should happen and a maximum buffer fullness $B_{\max}$ above which all incoming frames are dropped. The two buffer fullness levels $B_{\min}$, $B_{\max}$ and the corresponding dropping strategies lead to two extreme values for the Lagrange multipliers $\lambda_{\min}(n)$ and $\lambda_{\max}(n)$. The values for $\lambda(n)$ between $B_{\min}$ and $B_{\max}$ can be interpolated. We consider two different interpolation schemes for $\lambda(n)$ in this work.

Figure 3(a) illustrates a linear interpolation of $\lambda(n)$ between $\lambda_{\min}(n)$ and $\lambda_{\max}(n)$ as a function of the current buffer fullness $B(n)$. Hence, we write

$$\lambda(n) = \frac{B_{\max} - B(n)}{B_{\max} - B_{\min}} \cdot \lambda_{\min}(n) + \frac{B(n) - B_{\min}}{B_{\max} - B_{\min}} \cdot \lambda_{\max}(n). \tag{3}$$

Linear interpolation is the simplest way to interpolate $\lambda(n)$. An interpolation function that leads to more aggressive dropping if the buffer fullness approaches $B_{\max}$ can be realized by quadratic interpolation of $\lambda(n)$, as shown in Figure 3(b). With three control points A, B, and C, we can define a quadratic Bézier curve for $\lambda(n)$ with

$$\begin{aligned} A &= (A_x, A_y) = (B_{\min}, \lambda_{\min}(n)), \\ B &= (B_x, B_y) = (B_{\max}, \lambda_{\min}(n)), \\ C &= (C_x, C_y) = (B_{\max}, \lambda_{\max}(n)), \end{aligned} \tag{4}$$

$$P_x = (1-t)^2 \cdot A_x + 2t \cdot (1-t) \cdot B_x + t^2 \cdot C_x, \tag{5}$$

$$P_y = (1-t)^2 \cdot A_y + 2t \cdot (1-t) \cdot B_y + t^2 \cdot C_y. \tag{6}$$

The interpolated point $P = (P_x, P_y)$ moves on this curve from A to C by varying the parameter t from 0 to 1. For a given $B(n)$, we determine t and then $\lambda(n) = P_y$ from (6).

In order to determine $\lambda_{\min}(n)$, we evaluate (2) for every drop pattern and select $\lambda_{\min}(n)$ such that the minimum of (2) is obtained for the drop pattern where nothing is dropped in all K video streams. This means that

$$\begin{aligned} J_{p_n}(n) &= \sum_{k=1}^{K} \Delta D_{p_n}^k(n) - \lambda_{\min}(n) \sum_{k=1}^{K} \Delta R_{p_n}^k(n) \\ &\leq \sum_{k=1}^{K} \Delta D_p^k(n) - \lambda_{\min}(n) \sum_{k=1}^{K} \Delta R_p^k(n), \\ &\qquad \text{for } p \in P(n),\ p \neq p_n, \end{aligned} \tag{7}$$

where p_n represents the pattern when no frame dropping occurs in any of the video streams. As $J_{p_n}(n)$ equals zero, this leads to

$$\lambda_{\min}(n) \leq \frac{\sum_{k=1}^{K} \Delta D_p^k(n)}{\sum_{k=1}^{K} \Delta R_p^k(n)}, \quad \text{for } p \in P(n),\ p \neq p_n, \tag{8}$$

and we pick $\lambda_{\min}(n)$ to be as big as possible while still satisfying all the inequalities in (8). The value for $\lambda_{\max}(n)$ is derived in a similar fashion. For this, the minimization of (2) should now lead to the decision of dropping as many frames as possible (drop pattern p_a), which leads to

$$\begin{aligned} J_{p_a}(n) &= \sum_{k=1}^{K} \Delta D_{p_a}^{k}(n) - \lambda_{\max}(n) \sum_{k=1}^{K} \Delta R_{p_a}^{k}(n) \\ &\leq \sum_{k=1}^{K} \Delta D_{p}^{k}(a) - \lambda_{\max}(n) \sum_{k=1}^{K} \Delta R_{p}^{k}(n), \\ &\qquad \text{for } p \in P(n),\ p \neq p_a. \end{aligned} \tag{9}$$

This results in

$$\lambda_{\max}(n) \geq \frac{\sum_{k=1}^{K} \Delta D_{p_a}^{k}(n) - \Delta D_{p}^{k}(n)}{\sum_{k=1}^{K} \Delta R_{p_a}^{k}(n) - \Delta R_{p}^{k}(n)}, \quad \text{for } p \in P(n),\ p \neq p_a \tag{10}$$

and we pick $\lambda_{\max}(n)$ to be as small as possible while still satisfying all inequalities in (10).

3. FRAME DROPPING STRATEGIES FOR CONVERSATIONAL VIDEO

Compared with streaming video, conversational video is typically encoded in an IPPPPP... form. B-frames are normally not used because of the additional delay that would be introduced. Therefore, no "early" or even priority-based dropping as mentioned in Section 2.1 can be employed for conversational video. Video frames of multiple users are put into the buffer in a round robin (RR) way and dropped if the buffer cannot hold them. As conversational video does not have a GOP structure, the *distortion matrix* also cannot be used here to perform dropping decisions. Hence, we here propose to use the *hint tracks* [20, 21] as the side information and perform a utility-based frame dropping for conversational video to selectively drop the least important frames.

3.1. Side information for conversational video

Rate-distortion *hint tracks* are measured by feeding a specific loss pattern to the decoder and summing up the resulting increase in MSE over all affected frames of the video sequence. Without periodic I-frames in conversational video, there is no resynchronization between the encoder and decoder. Therefore, in order to increase the error resilience of the video stream to packet losses during transmission, slices (or rows) of macroblocks in video frames are intraupdated periodically, usually in a round-robin fashion. This is the so-called partial intraupdate. Figure 4 illustrates the error propagation when frame n is lost under the assumption that there is no remaining error propagating from earlier frames. The total distortion in this case is the sum of the distortions of all the following frames until the end of the video stream. However, with partial intraupdate, we can assume that the error propagation by the loss of frame n can be totally stopped after an equivalent intraupdate period of M frames.[3] Therefore, only the individual distortions up to frame $M + n$ need to be considered. Please note, here we calculate the *hint tracks* under the assumption that the losses of each frame are independent, which is the so-called zeroth-order distortion chain model DC^0 in [20]. This side information gives accurate distortion estimation when there is only one frame loss in the M consecutive frames. We can of course construct higher-order *hint tracks*, which can be extracted by feeding some loss patterns with more losses. However, high-order *hint tracks* have very high costs in terms of computational complexity as well as a huge storage requirement.

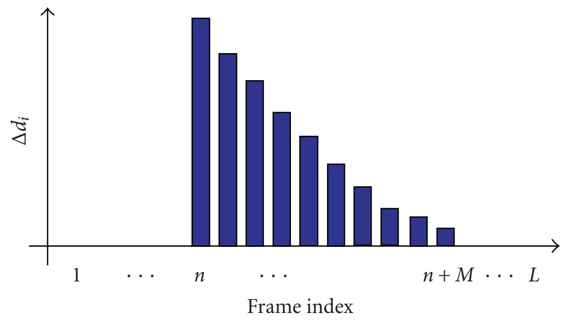


FIGURE 4: Error propagation for a single frame loss.

Since the future frame information for conversational video is unknown, it is impossible to premeasure the *hint track* (DC^0) value associated with a given loss pattern. Therefore, the model proposed in [22] is used to predict/estimate the distortion values $\Delta d(n + i)$ associated with future frames $n + i$ in the case of loss/drop of frame n. In particular,

$$\Delta d(n+i) = \begin{cases} \Delta d(n) \cdot r^i \cdot \left(1 - \dfrac{i}{M}\right), & \text{for } 0 \leq i \leq M, \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

where M is the equivalent intraupdate period, as explained above, and i indicates the distance between the future (concealed) frame and the lost frame n. In (11), $\Delta d(n)$ is the MSE information sent along with the video stream, representing the distortion of the current frame n in the case when this is the only lost frame and it is concealed by copying the previous frame. The attenuation factor r^i ($r < 1$) accounts for the effect of spatial filtering and the term $1 - i/M$ accounts for the intraupdate. Finally, the overall additional distortion $\Delta D(n)$ affecting the video sequence due to the loss of frame n, including error propagation into future frames, is then calculated as

$$\Delta D(n) = \sum_{i=0}^{M} \Delta d(n+i). \tag{12}$$

As a "copy and keep on decoding" error concealment scheme is employed in this case, the rate saving information

[3] This is the number of frames needed to intrarefresh all the macroblock locations in a video frame using this approach.

$\Delta R(n)$ associated with a given drop pattern is simply the sum of the size of the dropped frames. Therefore, only the sizes of the individual frames in bytes or bits need to be sent together with the *hint tracks* distortion information.

3.2. Utility-based frame dropping

Unlike streaming video, the importance of future frames in conversational video is unavailable. Therefore, it does not make sense to make dropping decisions for conversational videos until the buffer is unable to hold the new incoming frames. In particular, all new incoming frames are placed at the tail of the buffer queue if there is enough space left. Otherwise, we compare the importance of these frames and make a dropping decision.

In the *hint track* framework [18], for DC^0, the distortion ($\Delta D(n)$) and rate ($\Delta R(n)$) information associated with a video frame n comprise, respectively, the additional distortion affecting the reconstructed video sequence and the corresponding data rate reduction, when a single video frame n from the compressed video stream is dropped. The distortion-per-bit utility for a frame is then calculated as the ratio $\Delta D(n)/\Delta R(n)$ [20]. In our approach, we sort the current incoming (nth) frames from all K videos in decreasing order of their distortion per-bit utility. Then, from the head of the sorted list, frames are placed into the node's outgoing link buffer. If the frame at the current top of the list does not fit into the buffer, we turn to the next frame in the sorted list until no additional frame can be placed into the buffer. Please note, because of the tight delay constraint, optimization is done only among newly incoming video frames that correspond to a single time instant (one frame slot).

4. RD-OPTIMIZED DROPPING FOR STREAMING AND CONVERSATIONAL VIDEOS

In Sections 2 and 3, we have discussed the side information and dropping strategies for streaming and conversational videos. In this section, we consider the case when both types of video pass through the network node simultaneously and share one outgoing link.

4.1. Proposed framework

As shown in Figure 5, the RD-optimizer performs two independent dropping decisions for streaming video and conversational video, as proposed in [23]. The surviving (not being dropped) frames are stored in two independent classification buffers. The buffer for conversational video is relatively small in order to limit the forwarding delay experienced by these frames as this type of video application requires low latency. On the other hand, the classification buffer for streaming video is larger due to the more relaxed requirement on the delivery delay in this case. A scheduler is located behind the two buffers, which dynamically assigns the shared resource (forwarding data rate) to the two buffers by fetching video packets from them and putting them into the shared outgoing link buffer.

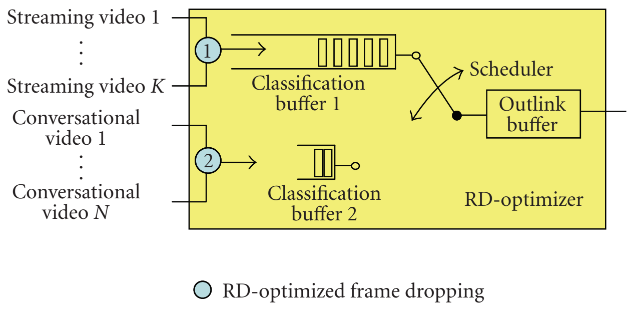


Figure 5: Structure of the RD-optimizer for frame dropping of streaming and conversational videos.

For streaming video, we opt to employ the cost function-based dropping strategies introduced in Section 2.3. The distortion-per-bit utility introduced in Section 3.2 is employed for the conversational video. For streaming video, information about future frames is taken into account. When the dropping decision is made for conversational videos, we can only compare the importance of the current frame with previous frames. As the selected frame is first put into the classification buffer, which we assume can be accessed by the RD-optimizer, frame replacement for this buffer is enabled. When new frames arrive at the node and the classification buffer is full, frames in the buffer with lower utility than the new incoming frame will be marked as dropping candidates. If the buffer space released by dropping these frames is enough to put in a new frame, they are physically dropped from the temporal buffer. On the other hand, if the released space is not enough to hold the new frame, it means the new frame is either too big or is not important enough for the reconstruction quality of the corresponding stream. Then this new frame is dropped and the marked frames in the buffer are recovered. Please note that this approach is equivalent to that taken in [20] for creating priorities among frames in a transmission window at a streaming server.

4.2. Scheduling strategies

Two separate classification buffers are employed to limit the additional delay experienced by the conversational video streams, as explained earlier. Compressed video has a variable bit-rate, and hence fixed resource assignment in terms of forwarding data rate sometimes wastes resources and leads to unnecessary frame dropping. With a dynamic resource assignment in place, the multiplexing of the multiple streams decreases the variation of the bit-rate and provides for more efficient resource utilization. Here, we propose two schemes for dynamic assignment of the data rate on the outgoing link.

4.2.1. Short-term mean-rate-based scheduling

Compressed video streams are typically VBR (variable bit rate), so when the outgoing link provides a transmission rate equal to the mean data rate of the incoming video stream, most likely some packets will be dropped if there is only a very small buffer at the node. But if we can perform the assignment adaptively following the variability of the stream's bit-rate, the node's forwarding resources can be more efficiently used. Without the knowledge of the sizes of future frames for conversational video, we can only make an estimate of the future bit rate, given the knowledge of the incoming data rate history. Here, we present a straightforward way to account for this. We take F past frames from each stream as an estimation window. The current resource assignment is then calculated as follows:

$$r_{\mathrm{SV}}^{i} = \sum_{k=1}^{K} \sum_{j=i-F-1}^{i-1} R_{j}^{k}, \tag{13}$$

$$r_{\mathrm{CV}}^{i} = \sum_{n=1}^{N} \sum_{j=i-F-1}^{i-1} R_{j}^{n}, \tag{14}$$

$$S_{\mathrm{SV}}^{i} = R_{\mathrm{out}} \cdot \frac{r_{\mathrm{SV}}^{i}}{r_{\mathrm{CV}}^{i} + r_{\mathrm{SV}}^{i}}, \tag{15}$$

$$S_{\mathrm{CV}}^{i} = R_{\mathrm{out}} - S_{\mathrm{SV}}^{i}. \tag{16}$$

In the equations above, r_{SV}^{i} and r_{CV}^{i} are the sum of bytes from the previous F frames of K streaming videos and N conversational videos, respectively. S_{SV}^{i} and S_{CV}^{i} represent the assigned transmission rate to the two buffers. R_{out} is the total transmission rate on the outgoing link and it is assumed to be constant during the whole transmission. With the same formulae (15) and (16), dynamic resource assignment for variable data rate on the outgoing link rate can also be accommodated by considering R_{out} to be a function of time.

4.2.2. *Buffer fullness-based scheduling*

Buffer fullness-based scheduling is an efficient way for the scheduler to avoid buffer overflow. When a buffer is heavily loaded, it means its incoming rate of traffic is bigger than the assigned service rate and therefore new incoming frames are likely to be dropped. In this case, a large portion of the outlink rate should be assigned to this buffer. On the other hand, when one of the two buffers is lightly loaded, it can still hold some new incoming frames. Hence, more transmission slots should be assigned to the other buffer then. Furthermore, when the two buffers have roughly the same fullness, it is not efficient to assign the same amount of resource to each of them, as their corresponding incoming rates may differ significantly. This is because the two buffers serve two different types of applications: streaming video and conversational video that usually have different data rates. Hence, we assign a weight to each buffer according to their incoming rates, and distribute the forwarding resource among them based on these weights.

The mean rates calculated with (13) and (14) represent the most recent (short-term) rates feeding the two buffers. Since they vary rapidly over time, employing them to determine the buffer weights may actually be inappropriate in this case. In particular, they may overly influence the resource allocation among the two buffers, thereby rendering their instantaneous fullness less important. Therefore, in order to avoid this effect we employ (17) and (18) instead which supply more stable cumulative mean rates.

The transmission rate assigned to the streaming videos at frame i can then be calculated with (19), and the remaining transmission capacity is assigned to the conversational videos,

$$r_{\mathrm{SV}}^{i} = \frac{1}{K\cdot(i-1)} \cdot \sum_{k=1}^{K} \sum_{j=1}^{i-1} R_{j}^{k}, \tag{17}$$

$$r_{\mathrm{CV}}^{i} = \frac{1}{N\cdot(i-1)} \cdot \sum_{n=1}^{N} \sum_{j=1}^{i-1} R_{j}^{n}, \tag{18}$$

$$S_{\mathrm{SV}}^{i} = R_{\mathrm{out}} \cdot \frac{r_{\mathrm{SV}}^{i}\cdot B_{\mathrm{SV}}^{i}}{r_{\mathrm{SV}}^{i}\cdot B_{\mathrm{SV}}^{i} + r_{\mathrm{CV}}^{i}\cdot B_{\mathrm{CV}}^{i}}. \tag{19}$$

Here r_{SV}^{i} and r_{CV}^{i} are, respectively, the mean incoming rates of the streaming videos and the conversational videos from the beginning until frame $i-1$. B_{SV}^{i} and B_{CV}^{i} denote, respectively, the fullness in percentage of the two buffers at the time instance when the ith frames of every stream arrive at the node.

5. COMPLEXITY ANALYSIS

In this section, we discuss the computational complexity and the storage requirements of the two RD-optimized frame dropping strategies proposed in this paper for streaming and conversational videos, respectively.

5.1. *Memory cost*

PRD/PRED are based on the static priority labels assigned to every frame, which are included in the bitstream, so there is no additional storage cost for PRD/PRED.

As shown in [19], the *distortion matrix* has $(1/2)N_G * (3 + N_G/(N_B + 1))$ entries for a GOP consisting of N_G frames with N_B B-frames between two P- or I-frames. However, less entries need to be stored in reality as in the cost function in (2), we only consider the overall (cumulative) additional distortion caused by selecting a dropping choice for a current frame. In particular, as explained in Section 2.3, there are at most four possible dropping decisions that can be made for each frame. Therefore, no more than four distortion values need to be associated to one video frame. Furthermore, given that the additional distortion is zero when nothing is dropped, there are only two remaining choices for which distortion values need to be stored in the case of P- and B-frames, and three such values in the case of I-frames. Hence, the distortion matrix can be compacted into $2*N_G+1$ entries for each GOP.

When the *hint track* framework based on the DC^0 distortion chain model is employed, frame drop patterns are constructed by considering every video frame independently [21]. Therefore, only L entries for a video stream with L frames need to be stored and sent as side information in the case of *hint track* DC^0. However, when higher-order distortion chain models are used in the *hint track* framework, the

Table 1: Construction of the test sequences.

Test sequence	Carphone	Claire	Foreman	Grandma	Miss America	Mother Daughter	Salesman	Suzie
# of frames	380	270	400	300	150	320	220	150
SV_1_20	5	1,4	3	—	2	6	—	—
SV_2_22	—	5,6	—	4	3	1	2	—
SV_3_24	—	4	2	6	3	1,5	—	—
SV_4_26	—	3	—	4	1	2,6	5	—
CV_1	1	3	4	—	—	5	—	2
CV_2	3	—	1	2,6	5	—	—	4
CV_3	—	2	—	3,6	—	1,5	4	—
CV_4	6	3	—	—	2	4	1,5	—

memory requirements are more demanding. In particular, the number of distortion values that need to be stored increases polynomially with the order of the distortion chain. For example, $L*(L-1)/2$ entries need to be stored in the case of *hint track* DC^1.

The rate information that needs to be stored is the same in both approaches and comprises the sizes of the video frames, as explained in Sections 2.2 and 3.1. Hence, there are L rate entries for L frames. Furthermore, for a given drop pattern, the associated rate reduction represents the sum of the sizes of the dropped frames in the case of the *hint track* framework, while for the *distortion matrix* approach, this quantity includes in addition the sizes of all dependent frames.

5.2. Computational complexity

The cost function-based frame dropping strategy for streaming video offers up to four possible dropping choices for every frame, which leads to an upper bound of 4^K drop patterns for K incoming streaming videos. As we need to calculate the distortion and rate saving for every drop pattern to select the optimal one, the computational complexity is very high in this case. However, in the cost function in (2) only one $\lambda(n)$ is used at every frame slot. For this reason, minimizing $J(n)$ is the same as minimizing $J_p^k(n)$ separately for each stream. Hence, we can rewrite the cost function as

$$\operatorname{argmin} J(n) = \sum_{k=1}^{K} \operatorname{argmin} J_p^k(n), \qquad \text{for } p \in P(n), \text{ for } p \in P^k(n) \tag{20}$$

so that the maximum number of possible drop patterns is reduced to $4*K$. Including the computation of $\lambda(n)$, the total calculation complexity is $O(8*K*L)$ for K videos, each of length L frames. Please note that this is the worst case that in practice is actually unattainable. That is because frame dropping decisions are only made when the buffer fullness reaches a predefined threshold. Furthermore, dropping decisions affecting future frames reduce the number of prospective drop patterns when the optimization is performed again, at the next frame slot.

With the utility-based approach for conversational video, the individual frames are considered independently for the DC^0 model. Therefore, there are only two possible dropping choices for every frame, to drop or not to drop and the resulting overall computational complexity is $O(N*\log(N)*L)$, where $N*\log(N)$ is the cost for sorting the importance at every frame slot. In particular, with the classification buffer in the hybrid scenario, assume that W frames are in the temporal buffer and need to be sorted according to their distortion-per-bit utility, the resulting computational complexity is $O((W+N)*\log(W+N)*L)$ in this case.

6. SIMULATION RESULTS

In this section, we examine the performance of several frame dropping strategies for streaming and conversational videos. First, we show the improvement achieved by the proposed RD-optimized frame dropping strategies introduced in Sections 2 and 3. Then, the performance of the frame dropping optimizer from Section 4 that considers both streaming and conversational videos is evaluated.

The videos employed in our simulation experiments are encoded with the H.264 MPEG-4/AVC codec [24] with a frame rate of 25 Hz. Long test sequences are generated by concatenating several short test sequences. For streaming video, each short sequence is appended at the tail of the resulting long sequence in integer multiples of the associated GOP length. This means that a number of frames at the end of a short sequence may be left out if its length is not an integer multiple of the GOP size. In Table 1, the entries in the first row under the names of the short sequences represent their corresponding lengths in number of frames. For example, the sequence *Carphone* is 380 frames long. Furthermore, the entries in each of the following rows, when moving towards the bottom of Table 1, represent the relative order of concatenation of the short sequences, for each of the resulting long sequences. For example, the long sequence SV_1_20 represents a concatenation of the short sequences: *Claire*, *Miss America*, *Foreman*, *Claire*, *Carphone*, *Mother*&*Daughter*, in this order. The test sequences are named SV_X_YY for streaming video, where YY stands for the length of the GOP and X is the index of the video. The number of B-frames between two P- or I-frames is set to be 1 in our experiments. For conversational

Table 2: Encoding characteristics of the test sequences.

Name	SV_1	SV_2	SV_3	SV_4	Sum/Avg
Rate (kbps)	92.44	67.99	69.55	50.60	280.58
Y-PSNR (dB)	38.63	38.32	38.10	38.24	38.33
Name	CV_1	CV_2	CV_3	CV_4	—
Rate (kbps)	122.07	119.06	67.81	116.42	425.36
Y-PSNR (dB)	37.57	37.26	37.36	37.69	37.47

videos, the name is CV_X. The encoding structure for conversational videos is IPPP... with an intraupdate interval of $M = 18$.

Table 2 summarizes the encoding (rate and quality) characteristics of the eight test sequences employed in our experiments. Furthermore, the entries in the last column in Table 2 represent, respectively, the sum of the mean rates and the average PSNR values for each of the two categories: streaming video and conversational video. As shown in Section 5.1, one GOP streaming video with N_G frames needs $2*N_G+1$ and N_G entries for the distortion and the rate information, respectively. With the assumption that each entry needs two bytes, each frame in SV_1 needs on average 6.1 bytes, which results in 0.152 kbps overhead traffic. Compared to the bitrate of the video stream at 92.44 kbps, this less than 0.2% overhead can be ignored. For the conversational video, the number of distortion entries is even smaller and compared to the bitrate of the video stream the overhead for the side information is insignificant.

In order to avoid the prospective loss of the very first I-frame for every test sequence, we assume that these frames have been forwarded by the network node and that all dropping decisions are made after the arrival of the second frame of each stream. For this reason, we set 7.5 KB out of 16 KB (total buffer size) as the initial buffer load in the case of streaming video when the frame dropping process starts. For conversational video, because of the strict delay constraint, the buffer size is set to be 5 KB. Again, the influence of the first I-frames is ignored and the initial buffer load is set to be 0 byte. The relation between the buffer size and the corresponding frame dropping performance and the decisions have been investigated in [23].

In our simulations, we measure the performance of a frame dropping strategy through the luminance (Y) PSNR values of the reconstructed video frames averaged over all videos. This quantity is computed as

$$\overline{\text{PSNR}} = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{L}\sum_{i=1}^{L}10\log_{10}\left(\frac{255^2}{\text{MSE}^k(i)}\right)\right), \qquad (21)$$

where K is the number of videos, each of length L frames, and $\text{MSE}^k(i)$ is the MSE distortion for frame i of video k.

6.1. Threshold settings for PRED

The implementation of PRED is straightforward and the only important point here is to select the proper thresholds for the random dropping of B-frames (T_1) and P-frames (T_2). In our experiments, the four streaming videos introduced in the previous section are employed as test videos. Note that no conversational videos are employed in the experiments, as no static priority labels can be established for them ahead of time. The operation of PRED on such content reduces to random dropping without priorities. In our experiments here, we go through all the possible values for T_1 from 30% to 100% of the buffer fullness and T_2 is always bigger than or equal to T_1.

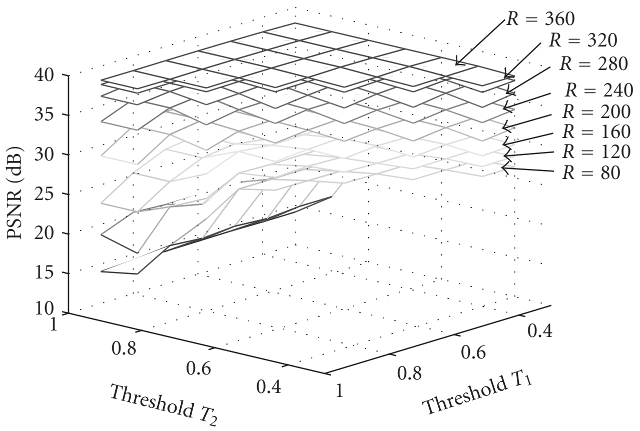


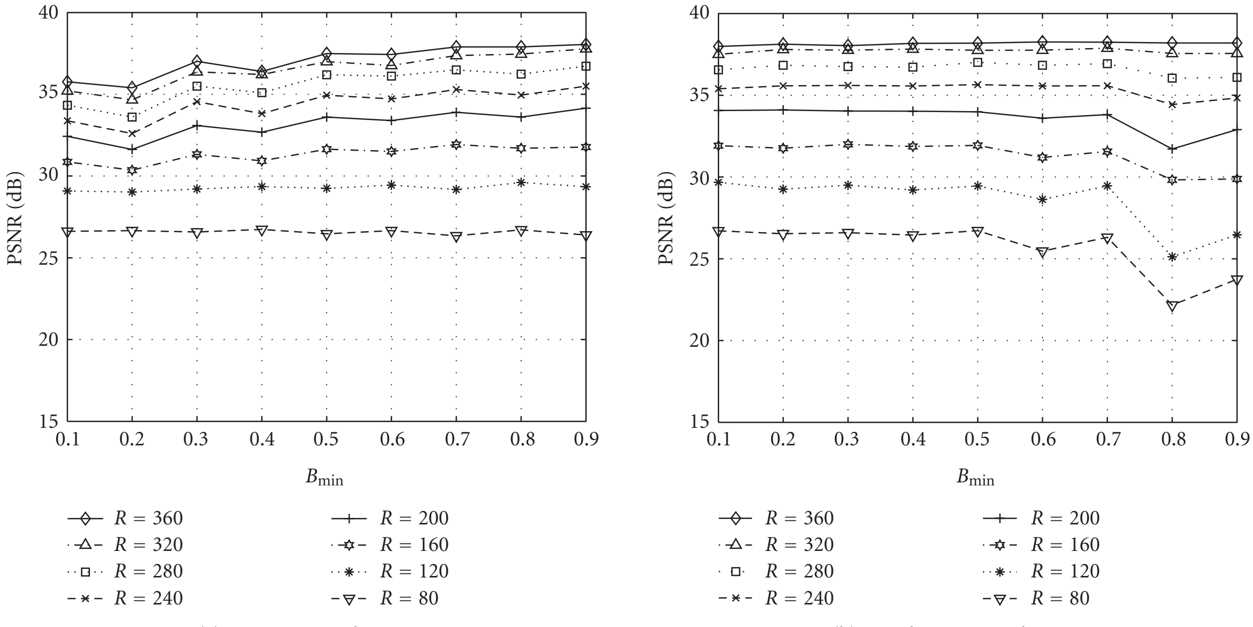
Figure 6: Setting of PRED thresholds, R represents the outlink rate in kbps.

In Figure 6, we show the average reconstruction quality (Y-PSNR) of the four streaming videos as a function of T_1 and T_2 at different outlink transmission rates, which are represented with different surfaces in the figure. In principle, the higher the transmission rate, the higher the reconstruction video quality. However, we can see that at low rates, the performance surface is not flat and a big performance drop can be observed when large values for T_1 and T_2 are selected. The performance at higher rates is more stable, as the observed reduction in video quality due to an improper selection of thresholds does not exceed 1~1.5 dB here. The upper and lower performance bounds of PRED are shown in Table 3, which are the highest and lowest points on each surface in Figure 6. The normalized rate is the percentage of available transmission rate versus the mean rate of all users. We can see from the table that a large performance gap exists between the two bounds for the case when the transmission rate on the forwarding link is much smaller than the mean aggregate source rate of the videos. However, it is not easy in practice always to select the optimal thresholds such that the upper performance bound is achieved.

As described in Section 2.1, we can have different dropping thresholds for P-frames depending on their position in a GOP. For example, if we set the start point for dropping frames to be at 50% of the buffer size, and then each successive dropping threshold to be associated with a further increment of 5%, we achieve the upper performance bound for the case when only two thresholds are used. This means that more accurate frame dropping decisions can be made, when finer priority steps in terms of frame dropping are employed.

Table 3: Performance bounds of PRED.

Rate (kbps)	80	120	160	200	240	280	320	360
Normalized rate	0.285	0.428	0.570	0.713	0.855	0.998	1.140	1.283
$\text{PSNR}_{\max}$ (dB)	26.43	27.75	29.37	31.91	35.18	36.83	37.82	38.23
$\text{PSNR}_{\min}$ (dB)	14.03	16.67	21.68	28.17	32.77	34.73	36.50	37.47



(a) Linear interpolation
(b) Quadratic interpolation

Figure 7: Performance of DM-based frame dropping, R represents the outlink rate in kbps.

6.2. *Cost function-based RD-optimized frame dropping*

In the following, we examine the influence of λ on the performance of cost function-based frame dropping for the two interpolation methods introduced in Section 2.3. In Figures 7(a) and 7(b), we show the results for the cost function-based frame dropping strategy when using linear and quadratic interpolations for the multiplier λ, respectively. In all simulations, we fix $B_{\max}$ to be 100% of the buffer size.

Quadratic interpolation exhibits a degraded quality when very high values for $B_{\min}$ are selected at very low outlink rates, as shown in Figure 7(b). This is because quadratic interpolation leads to aggressive frame dropping decisions when the buffer fullness approaches $B_{\max}$ and is far away from $B_{\min}$. Setting $B_{\min}$ to be bigger than 0.8 results in late dropping of less important frames and which in turn causes unnecessary loss of some frames with high importance. The curves are smooth and flat when $B_{\min}$ is smaller, as the dropping decision is very moderate when the buffer is lightly loaded. When linear interpolation is used, small values for $B_{\min}$ at high outlink rates lead to unnecessary dropping of some frames with low importance. To summarize, selecting $B_{\min}$ larger than 0.5 is fine for linear interpolation and for quadratic interpolation, $B_{\min}$ should be selected smaller than 0.6. By selecting $B_{\min}$ between 0.5 and 0.6, we obtain good results for both schemes.

6.3. *Performance comparison among all frame dropping schemes for streaming video*

In this section, we compare the performance of the frame dropping schemes for streaming video examined in this paper, as a function of the forwarding data rate on the outgoing link. In Figure 8, PRD denotes the priority-based random frame dropping. PRED here fixes the thresholds T_1 and T_2 to be 70% and 90%, respectively, of the buffer fullness, while the PRED_UB curve in Figure 8 corresponds to the upper bound from Table 3. Our proposed RD-optimized cost function-based dropping strategy that uses the *distortion matrix* as the side information is shown as the CF_DM curve in the figure, where $B_{\min}$ is selected to be 0.6.

PRD performs the worst at all the rates, as can be seen from Figure 8. PRED also shows a poor performance at low link rates, while PRED_UB performs much better by the proper selection of dropping thresholds. CF_DM outperforms all other schemes as a result of its accurate distortion estimation and dynamic adjustment of the dropping aggressiveness according to the buffer fullness level.

6.4. *Utility-based frame dropping for conversational video*

We compare our utility-based frame dropping for conversational video with the pure random dropping in a round robin fashion. When a video packet arrives, if the outgoing link buffer can still hold it, the packet is put into the buffer, otherwise, this packet is simply dropped. For the utility-based approach, when N new incoming frames arrive at the node and the buffer cannot hold all of them, they are sorted according to their utility and put into the buffer one after another until the buffer is full.

Table 4 shows the averaged PSNR values of the luminance (Y) component for the four test conversational videos at different outgoing link rates. The mean score of the four videos (boldfaced numbers) presents the overall reconstruction quality. The utility-based frame dropping outperforms the random frame dropping in the range of middle-to-high rates, because at very low rates, consecutively dropping of a large number of frames leads to an inaccurate estimation of distortion. However, if we look at the performance of individual users, it is more fair by using the utility-based approach (maximum difference from 0.9 dB~5.5 dB) compared to the pure random dropping approach (maximum difference from 3.8 dB~10 dB). Therefore, in addition to the overall quality, our approach also shows a good characteristic with respect to the fairness among users.

6.5. *Joint optimization for streaming and conversational videos*

In this experiment, we compare our joint optimizer for streaming and conversational videos with a reference scheme that uses PRED/RR for these two types of video applications, respectively. In particular, streaming video provides three types of static priority labels, as explained earlier. Therefore, in the reference scheme we can perform PRED on the streaming videos by early dropping of B- or P-frames. On the other hand, for conversational video, all the frames, except the very first one, are P-frames and hence there is no static priority difference among them. Therefore, when multiple frames arrive simultaneously at the network node, a simple round-robin scheme (over the conversational videos to which these frames belong) is employed to determine how many of them can be placed into the corresponding buffer for conversational video.

Our proposed optimizer uses the *distortion matrix* for streaming video and *hint tracks* for conversational video. In the case of conversational video, we employ (11) and (12) to estimate the overall distortion associated with the dropping of a single frame, as explained in Section 3.1. In the equations, the equivalent intraupdate period M is set to be 18 frames and the attenuation factor r is set to be 0.997. Finally, for comparison purposes we also consider the hypothetical

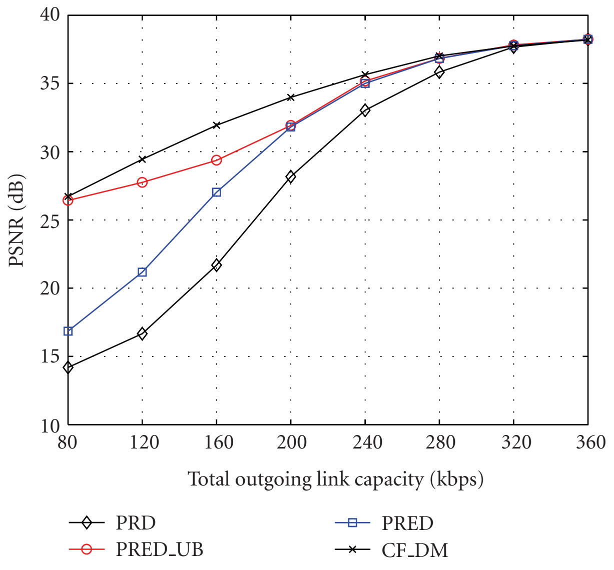


FIGURE 8: Performance comparison of different frame dropping schemes for streaming video.

case when the distortion incurred by dropping frames can also be precalculated for conversational video.

Figure 9 shows the performance improvement achieved by the proposed RD-optimized strategy for dropping frames from both streaming and conversational videos. Several instances of the proposed optimizer are considered in Figure 9. In particular, RD_FIX denotes the proposed optimizer with fixed resource assignment of 40% to the streaming videos and 60% to the conversational videos. Note that this assignment corresponds to the overall average data rates for these two types of videos. Furthermore, RD_BUF and RD_RAT in the figure represent the buffer fullness-based and the short-term mean-rate-based scheduling strategies introduced in Section 4.2, respectively. In the case of RD_RAT, F in (13) and (14) is set to be 10 frame slots in this experiment.

First, it can be seen that when the outgoing link rate is larger than the mean $B_{\min}$ incoming rate (when the normalized rate is larger than 1), the performances of the RD-optimizer and PRED/RR are similar. However, there is still a performance improvement of 1 dB at 900 kbps. This is because even at this rate, frame dropping from the conversational videos need to occur in PRED/RR, whenever the incoming data rate of the video streams peaks, as the small buffer for conversational videos cannot hold too many frames at once. The RD-optimizer deals more successfully with this situation, since the optimized frame dropping has more opportunities to drop the least important frames even if they have been in the classification buffer waiting to be scheduled. At the same time, the dynamic resource assignment saves away some spare transmission slots from the streaming videos, that can be appropriately reallocated to the conversational videos afterwards, as explained in Section 4.2. When the outgoing rate is smaller than the total traffic rate, an improvement of around 3 dB is observed, as shown in Figure 9.

Table 4: Comparison of utility-based dropping and random dropping.

Outlink rate (kbps)	120	180	240	300	360	420	480	540
Normalized rate	0.282	0.423	0.564	0.705	0.846	0.987	1.128	1.270
Utility-based frame dropping								
CV1(dB)	15.01	19.60	22.53	27.23	29.96	32.20	34.01	36.07
CV2(dB)	15.60	21.07	23.30	28.46	31.11	33.03	35.12	36.45
CV3(dB)	20.50	21.30	23.34	27.23	30.39	32.76	34.25	36.06
CV4(dB)	18.52	21.37	25.12	27.83	31.40	34.02	35.82	36.93
Mean(dB)	**17.41**	**20.83**	**23.57**	**27.69**	**30.71**	**33.00**	**34.80**	**36.38**
Random frame dropping								
CV1(dB)	13.68	19.96	21.09	25.25	26.57	29.75	31.71	35.17
CV2(dB)	16.40	16.64	21.15	23.52	28.93	28.60	34.43	33.34
CV3(dB)	16.44	26.26	26.25	29.17	31.69	34.99	34.11	37.02
CV4(dB)	23.67	24.49	28.75	29.97	32.61	33.84	36.38	37.21
Mean(dB)	**17.55**	**21.84**	**24.31**	**26.98**	**29.95**	**31.79**	**34.16**	**35.69**

Table 5: Assignment of forwarding date rate.

Total_rate (kbps)	200	300	400	500	600	700	800	900
Normalized rate	0.283	0.425	0.567	0.708	0.850	0.992	1.133	1.275
Assigned_rate_SV (kbps)	72.86	104.82	140.40	180.10	227.64	258.44	272.88	279.18
(%)	(0.36)	(0.35)	(0.35)	(0.36)	(0.38)	(0.37)	(0.34)	(0.31)
Assigned_rate_CV (kbps)	127.14	195.18	259.60	319.90	372.36	441.56	527.12	620.82
(%)	(0.64)	(0.65)	(0.65)	(0.64)	(0.62)	(0.63)	(0.66)	(0.69)

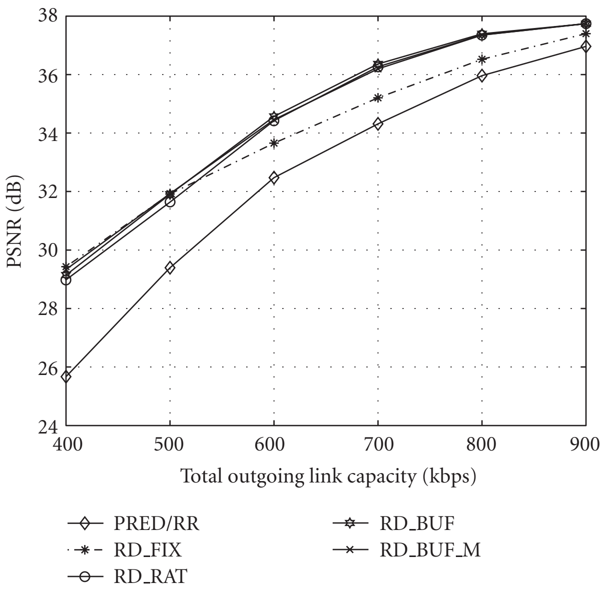


Figure 9: Performance comparison of the proposed RD-optimizer and PRED/RR for streaming and conversational videos.

Furthermore, at low rates, the performances of RD_FIX, RD_BUF, and RD_RAT are almost the same. However, at high rates the schemes with dynamic resource assignment perform much better, because reassigning some of the resources from the streaming video buffer to the conversational video buffer will not influence the quality of streaming video significantly, as these resources are typically saved when the low data rate sections of the incoming streams occur at the node. But with fixed resource allocation, these unused resources from the streaming video are wasted, which leads to degraded performance at high outgoing link rates compared to the case of dynamic resource assignment. Table 5 gives the assigned transmission resources to the streaming videos and conversational videos when the buffer fullness-based scheduling strategy is used. More transmission resources are assigned to the conversational videos compared to their mean bitrates. This is the consequence of the small size of the classification buffer due to the tight delay constraint of the conversational videos.

Finally, precomputed *hint tracks* for conversational video are not available in practice, but here we compute them anyhow in order to examine if the approximation from (11) and (12) leads to accurate results. Our experiments show that precomputed *hint tracks* (RD_BUF_M) for the conversational videos and the approximation (RD_BUF) obtained using (11) and (12) lead to almost identical performance results, as can be seen from Figure 9. The estimation bias from the model in (11) and (12) does not affect the results, because the relative values of the distortion-per-bit utility among the individual frames are preserved in either case.

7. CONCLUSIONS

We have presented RD-optimized frame dropping strategies for streaming and conversational video applications that can be applied at active network nodes. The proposed techniques employ side information about the packetized video content that is extracted at compression time and that is sent along the video streams. The only additional information that the techniques need to operate at an active node is the fullness of the outlink buffer and the mean traffic rate of the video streams passing through the node. It is shown through simulations that a significant improvement in video quality is achieved over previous approaches, by a judicious selection of side information and optimized frame dropping strategy.

ACKNOWLEDGMENT

This work has been supported in part by DFG Grant STE 1093/3-1.

REFERENCES

[1] A. Vetro, C. Christopoulos, and H. Sun, “Video transcoding architectures and techniques: an overview,” *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, 2003.

[2] J. Xin, C.-W. Lin, and M.-T. Sun, “Digital video transcoding,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, 2005.

[3] A. Vetro, J. Cai, and C. W. Chen, “Rate-reduction transcoding design for wireless video streaming,” *Journal of Wireless Communications and Mobile Computing*, vol. 2, no. 6, pp. 625–641, 2002.

[4] W. Feng, M. Liu, B. Krishnaswami, and A. Prabhudev, “A priority-based technique for the best-effort delivery of stored video,” in *Multimedia Computing and Networking*, vol. 3654 of *Proceedings of SPIE*, pp. 286–300, December 1999.

[5] Y. Lu and K. J. Christensen, “Using selective discard to improve real-time video quality on an ethernet local area network,” *International Journal of Network Management*, vol. 9, no. 2, pp. 106–117, 1999.

[6] H. Cha, J. Oh, and R. Ha, “Dynamic frame dropping for bandwidth control in MPEG streaming system,” *Multimedia Tools and Applications*, vol. 19, no. 2, pp. 155–178, 2003.

[7] R. Mahajan, S. Floyd, and D. Wetherall, “Controlling high-bandwidth flows at the congested router,” in *Proceedings of IEEE International Conference on Network Protocols (ICNP ’01)*, pp. 192–201, Riverside, Calif, USA, November 2001.

[8] J. Kritzner, M. Kampmann, and U. Horn, “Comparison of frame dropping strategies for adaptive video streaming,” in *Proceedings of International Picture Coding Symposium (PCS ’04)*, pp. 325–329, San Francisco, Calif, USA, December 2004.

[9] J. Kritzner, U. Horn, and M. Kampmann, “Priority generation for video streaming using stream decodability,” in *Proceedings of the 14th International Packet Video Workshop (PV ’04)*, Irvine, Calif, USA, December 2004.

[10] P. A. Chou and Z. Miao, “Rate-distortion optimized streaming of packetized media,” Tech. Rep. MSR-TR-2001-35, Microsoft Research, Mountain View, Calif, USA, February 2001.

[11] F. Zhai, R. Berry, T. N. Pappas, and A. K. Katsaggelos, “A rate-distortion optimized error control scheme for scalable video streaming over the internet,” in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME ’03)*, Baltimore, Md, USA, July 2003.

[12] G. J. Sullivan and T. Wiegand, “Rate-distortion optimization for video compression,” *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[13] H. Schwarz and T. Wiegand, “An improved MPEG-4 coder using lagrangian coder control,” in *Proceedings of the 4th International ITG Conference on Source and Channel Coding (SCC ’02)*, Berlin, Germany, January 2002.

[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 Asilomar Conference on Signals, Systems and Computers (ASILOMAR ’01)*, vol. 1, pp. 210–214, Monterey, Calif, USA, November 2001.

[15] E. Masala, H. Yang, K. Rose, and J. C. De Martin, “Rate-distortion optimized slicing, packetization and coding for error resilient video transmission,” in *Proceedings of IEEE Data Compression Conference (DCC ’04)*, pp. 182–191, Snowbird, Utah, USA, March 2004.

[16] I. Bouazizi, “Size-distortion optimized proxy caching for robust transmission of MPEG-4 video,” in *Proceedings of International Workshop on Multimedia Interactive Protocols and Systems (MIPS ’03)*, no. 2899, pp. 131–142, Napoli, Italy, November 2003.

[17] P. Yin, A. Vetro, M. Xia, and B. Liu, “Rate-distortion models for video transcoding,” in *Proceedings of Conference on Image and Video Communications and Processing*, vol. 5022 of *Proceedings of SPIE*, pp. 479–488, Berlin, Germany, May 2003.

[18] J. Chakareski and P. Frossard, “Rate-distortion optimized bandwidth adaptation for distributed media delivery,” in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME ’05)*, vol. 2005, pp. 763–766, Amsterdam, Netherlands, July 2005.

[19] W. Tu, W. Kellerer, and E. Steinbach, “Rate-distortion optimized video frame dropping on active network nodes,” in *Proceedings of the 14th International Packet Video Workshop (PV ’04)*, Irvine, Calif, USA, December 2004.

[20] J. Chakareski, J. Apostolopoulos, S. Wee, W.-T. Tan, and B. Girod, “R-D hint tracks for low-complexity R-D optimized video streaming,” in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME ’04)*, vol. 2, pp. 1387–1390, Taipei, Taiwan, June 2004.

[21] J. Chakareski, J. G. Apostolopoulos, S. Wee, W.-T. Tan, and B. Girod, “Rate-distortion hint tracks for adaptive video streaming,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1257–1269, June 2005.

[22] Y. J. Liang, J. G. Apostolopoulos, and B. Girod, “Analysis of packet loss for compressed video: does burst-length matter?” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’03)*, vol. 5, pp. 684–687, Hong kong, April 2003.

[23] W. Tu, J. Chakareski, and E. Steinbach, “Rate-distortion optimized frame dropping and scheduling for multi-user conversational and streaming video,” in *Proceedings of the 16th International Packet Video Workshop (PV ’06)*, Hangzhou, China, April 2006.

[24] HHI. The JVT H.264/MPEG-4 AVC reference software. http://iphome.hhi.de/suehring/tml/index.htm.