Available online at www.sciencedirect.com
ScienceDirect
Signal Processing: *Image Communication* 22 (2007) 374–388
SIGNAL PROCESSING: IMAGE COMMUNICATION
www.elsevier.com/locate/image

Real-time and rate–distortion optimized video streaming with TCP[☆]

Antonios Argyriou

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

Received 5 February 2006; received in revised form 30 December 2006; accepted 8 January 2007

Abstract

In this paper we explore the use of a new rate–distortion metric for optimizing real-time Internet video streaming with the transmission control protocol (TCP). We lay out the groundwork by developing a simple model that characterizes the expected latency for packets send with TCP-Reno. Subsequently, we develop an analytical model of the expected video distortion at the decoder with respect to the expected latency for TCP, the packetization mechanism, and the error-concealment method used at the decoder. Characterizing the duo protocol/channel more accurately, we obtain a better estimate of the expected distortion and the available channel rate. This better knowledge is exploited with the design of a new algorithm for rate–distortion optimized encoding mode selection for video streaming with TCP. Experimental results for real-time video streaming depict improvement in PSNR in the range of 2 dB over metrics that do not consider the behavior of the transport protocol.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Video streaming; TCP; Distortion model

1. Introduction

Streaming pre-encoded or real-time video over the Internet is a popular application that continues to gain interest as new entertainment services are continuously introduced. In today's Internet, the transmission control protocol (TCP) protocol that transports the bulk of the existing traffic [8] is considered unsuitable for video streaming applications, while its counterpart UDP is usually the protocol of choice. The main reasons why TCP is unsuitable for this class of applications are the rapid throughput fluctuations and the reliability mechanism which introduces additional delays [31]. Therefore, it is generally believed that the transport protocol of choice for video streaming should be UDP, on top of which several application-specific mechanisms can be built (error control, rate control, etc.) [31]. However, the absence of congestion control from UDP can cause performance deterioration for TCP-based applications if wide-scale deployment takes place in the Internet [8,14]. That is the reason behind the IETF effort to define a new rate control protocol, which is characterized by the same behavior with TCP in the long run, but allows smoother throughput fluctuations [12]. Even so, this standardization effort is in progress and is considered an active research topic. Despite these

[☆]A short version of this paper appeared in ICME 2006.
E-mail addresses: anargyr@ece.gatech.edu, anargyr@ieee.org (A. Argyriou).

efforts, however, the majority of commercial IP-based video streaming systems unexpectedly so employ TCP for transport layer services of pre-encoded video content [25,32,24]. The widespread use of TCP and its well-understood behavior dominated over the concerns regarding TCP's deficiencies.

We will see that even with the reliable TCP, the problem is essentially error control. Handling errors is a critical task in a video communication system. Especially for real-time video communications delay constraints are very strict, making thus retransmission of lost packets not particularly useful in a practical setup. To overcome this problem, methods like forward error correction (FEC) are usually employed [26,9]. However, the primary environment where FEC methods are effective is when random bit errors hit the bitstream, while buffer overflow in the Internet generates packet erasures because of the packet drops. This is what would happen if we build a video streaming system with TCP. Furthermore, even though TCP has the inherent retransmission mechanism for error control, when the application payload are video packets, error control obtains a new meaning. This is true because TCP may be able to recover packets with retransmissions but they may be received too late for playback. While someone could "hack" TCP so that it can avoid unnecessary retransmissions, we believe that this cross-layer option is rather invasive to the standardized TCP.

Therefore, blindly using TCP for video streaming will not yield the best possible performance. We believe that when TCP is used for video streaming, we should take into consideration its behavior explicitly so that we can maximize the delivered video quality. The question that we will answer in this paper is how to perform this task efficiently. TCP is a reliable protocol and uses retransmission for packets assumed lost, according to its internal mechanisms. These mechanisms are fast retransmission and retransmission after timer expiration. These mechanisms incur a specific latency for the retransmitted packets which can be calculated precisely and captured in a simple model. In this way the encoder can take into account the additional delays (which are translated to lost macroblocks) explicitly and in real-time. The encoder can also use the closed-form TCP throughput formula for rate control [23].

Our intention with this paper is to demonstrate that real-time video encoding and TCP streaming can be done more efficiently together than the current practice, since we can tackle some of the problems by employing a new systematic optimization approach. What we suggest is to use a new rate–distortion metric that takes into account the aforementioned TCP behavior (through the developed models), so that we can estimate the expected distortion of the video signal at the decoder. Subsequently, the real-time encoder will use this metric in order to optimize decisions for the encoding of individual macroblocks. This is essentially an implicit cross-layer mechanism since the application will regulate its behavior according to the limitations of the underlying protocol but without requiring any modifications of the protocol stack. In summary, two are the goals that we set to achieve with this paper:

- Derive analytically the expected distortion for a video bitstream at the decoder when TCP-Reno is used for transport.
- Select optimal encoding mode for each individual macroblock at the encoder.

The rest of this paper is organized as follows. In Section 2 we present the related work in the area of video streaming with TCP. Section 3 presents an overview of the proposed system, while important aspects of our system like the used codec, packetization method, error-concealment method, and network model are covered in more detail. In Section 4 we analyze, based on the previous configuration, how specific encoded macroblocks of a video sequence are sent and lost with TCP. From this analysis we derive the macroblock loss probability as a function of TCP sender parameters. Subsequently, Section 5 analyzes how the latency of these lost macroblocks is affected, since they are retransmitted by the reliable TCP protocol. By knowing which macroblocks are lost, and if they can meet their deadline or not, we present in Section 6 an analytical model of the expected decoder distortion. In Section 7, we present an algorithm for the main goal of our analysis: select the optimal encoding mode based on the expected decoder distortion. Experimental results are finally presented in Section 9 while Section 10 concludes this paper.

2. Related work

During the last few years, the research community has proposed a number of optimization mechanisms for video streaming with TCP. We review the most noteworthy of these mechanisms next. One of the first mechanisms is time-lined TCP [22], where TCP video streaming is realized by allowing the operating system to control the transmission of data that have strict deadlines. A similar approach to the previous one can be found in [17]. In that work, the main task of the TCP-RTM receiver is to deliver out-of-order packets to the application regardless of when they arrive. But this approach requires modifications both to the TCP sender and receiver. However, both the aforementioned approaches do not deal with issues related to the TCP retransmissions and bandwidth fluctuations. In a receiver oriented approach [16], the authors evaluate multimedia streaming with TCP, and conclude that buffering at the client can handle the retransmission delays and the congestion control induced throughput variations of TCP. Another receiver-driven technique for video streaming, that introduced the idea of receiver-based delay control (RDC), was presented in [13]. The idea is that a receiver delays TCP acknowledgments based on feedback from routers. One interesting mechanism for TCP-based streaming can be found in [19]. The main idea that the authors had was to provide an approximate CBR channel to the streaming flow that is using TCP, through prioritization over other flows at the last mile connection of the receiver. This is the bandwidth sharing system (BWSS). An analytical study that we are aware of has been reported in [30], and the authors present a model for a simple TCP-based video streaming scenario. One of the most recent works can be found in [10], where the proposed technique for TCP streaming is realized at the server which selectively drops frames in order to match the available bandwidth.

In summary, one common characteristic of majority of the aforementioned mechanisms is that they consider modifications and enhancements to TCP or the infrastructure, while they ignore the nature of the content which is a video bitstream. Mechanisms that do consider the nature of video data are the source coding network adaptive algorithms [31]. For example, an interesting approach is rate–distortion optimization (RDO) [18], which is based on the tradeoffs between quality and bitstream size. The advantage of this approach is that it can accommodate rate fluctuations with appropriate modification of the quantization parameter at the encoder. However, large variations in the quality of the received video and significant processing overhead are two of its disadvantages. Other approaches use RD metrics in order to select the optimal encoding mode at the macroblock level at the sender [36,33,6]. A complete and general framework for the RD optimized packet scheduling has been developed in [5,4], where the authors estimate the end-to-end video distortion for several streaming scenarios. They propose a heuristic algorithm for finding a suboptimal scheduling policy that optimizes the receiver distortion. Several more sophisticated algorithms have been developed since then [3,15,35]. In [35], a new scheme for RD optimized streaming for MPEG-4 layered video considers the use of unequal error protection through FEC. Other works for RD optimized streaming include [20], in which the authors proposed new heuristic algorithms for estimating the expected distortion in real-time. While most of the above RD-based algorithms are fairly sophisticated, they ignore one important parameter which is the behavior of the protocol stack.

3. System overview

Fig. 1 depicts a simplified block diagram of the end-to-end real-time video streaming system. Based on this system model we will derive the desired metric that captures the end-to-end distortion. Besides encoding the input sequence, the encoder performs rate control by polling the TCP protocol in order to obtain the instantaneous available channel rate. At the client, a small startup delay is used before the video playback starts, while the server continuously sends video packets which are stored at the playback buffer. In our system, this startup delay depends on the number of correctly received frames, which should be more than eight before playback commences. We have to note that since we are using the connection-oriented TCP, the duration of the startup delay is slightly longer.

While the above sequence of events is typical in a client/server type of streaming application, a set of extra steps are performed by our system. Throughout the streaming session, the streaming server estimates in real-time the end-to-end distortion as a function of the distortion of the individual macroblocks, the expected end-to-end latency, the packetization method, and the error-concealment method at the decoder. Besides the aforementioned parameters that are estimates, the parameters that

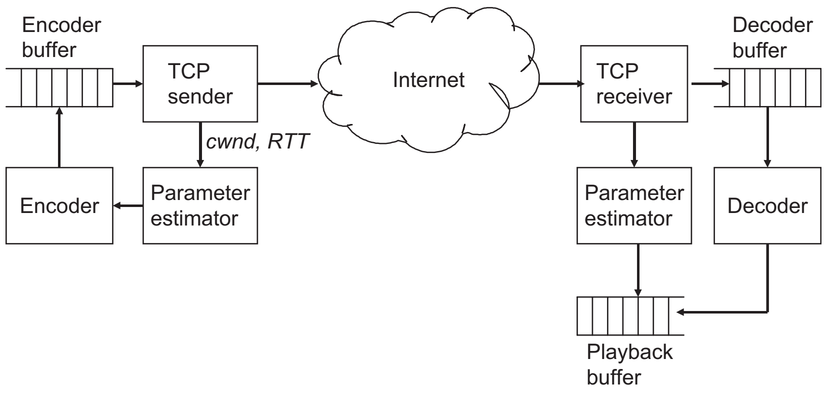

Fig. 1. Proposed media streaming architecture based on TCP.

are used and are based on actual measurements are the value of the congestion window and the phase of the TCP sender (slow start or congestion avoidance).

This cross-layer information is used for enhancing the short-term projections of the available TCP rate. According to the algorithm that we will present, the real-time encoder selects an optimal encoding mode for each individual macroblock based on the RD metric that was just described. Based on the expected distortion, we use Lagrangian relaxation to find the optimal encoding mode for each macroblock which might be intra or inter. Intraencoding means that spatial redundancy is exploited at the encoder (with DCT), while the intermode means that temporal redundancy is also exploited (motion estimation/compensation). Therefore, we can understand that estimating the expected distortion timely and accurately should be one of the primary concerns of this work. The explanation of specific encoder options follows next.

3.1. Coding options and packetization

The input video bitstream is compressed with the H.263++ encoder [11] and for rate control we maintain the TMN8 implementation. The selection of the packetization method at the sender is important, since it can have different effects on the quality of the reconstructed video at the decoder. Therefore, we investigate two error-resilient packetization methods that are based on a slice organized bitstream. These methods are scattered slices and macroblock row interleaving (Fig. 2). The use of a slice-structured bitstream is an error-resilient tool that works in close cooperation with

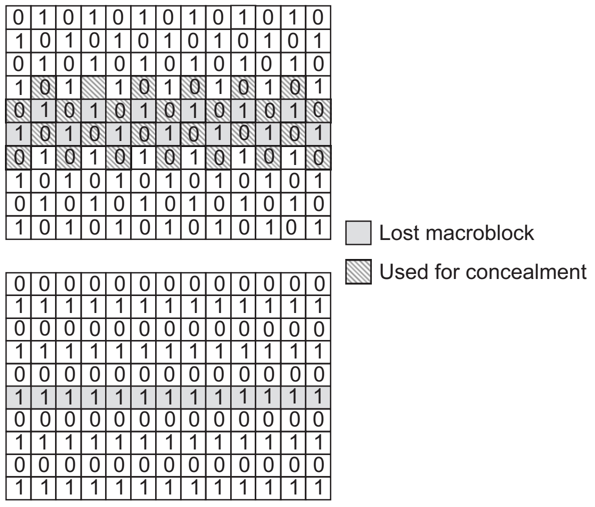

Fig. 2. Packetization of encoded macroblocks according to the scattered slices and row interleaving methods.

the error-concealment at the decoder, which will also be considered as part of our distortion model. However, one problem with any interleaved packetization method is that the source coder will perform block level prediction only within the blocks that will be packetized sequentially. This approach results in a slight reduction of the prediction gain.

Now, in Fig. 3 we see in detail the adopted packetization method for TCP. Each frame is split into several macroblocks which are grouped into slices. A TCP packet probably carries more than one slice. In each of these slices, the appropriate RTP headers are inserted. It is possible, however, that a group of macroblocks that belong to a frame do not fit exactly into one TCP packet.

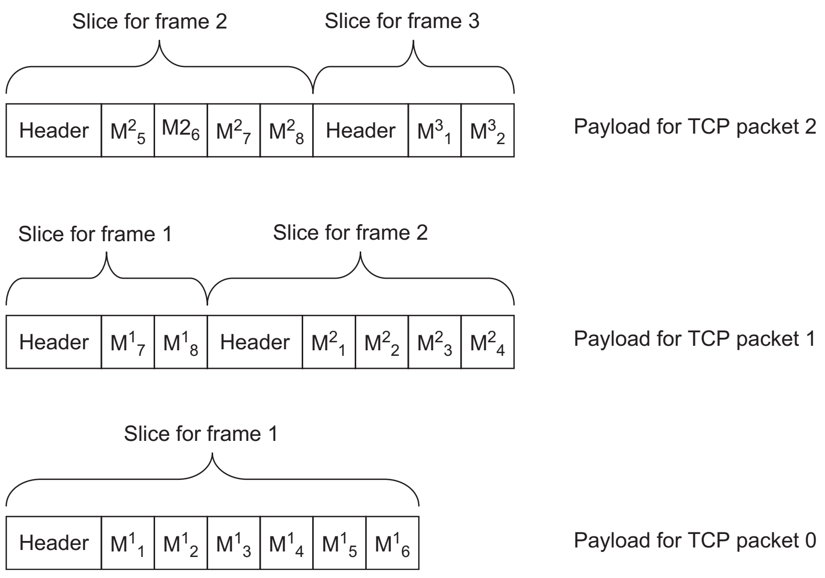

Fig. 3. Proposed packetization method.

The macroblocks that do not fit form a separate slice and are placed in the next TCP packet.

A simple hybrid error-concealment method is incorporated into the model. The first one (EC1) uses temporal interpolation, and replaces a lost macroblock with the same macroblock from the previous frame. The second (EC2) is an enhanced version of EC1 and copies the motion vector from a correctly received neighboring macroblock, and identifies the appropriate macroblock from the previous frame. Intuitively, we can see that when the packet size allows only a small number of MBs, then EC2 would perform better since many adjacent macroblocks will be available in the face of a packet loss. In our system the concealment method is selected during initialization of the streaming session.

3.2. Channel model

Many studies have shown that the first-order two-state Markov chain (i.e. the Gilbert path model) can approximate fairly well the behavior of the Internet. However, in this paper we adopt the Bernoulli path model which is a simplification of the Gilbert model. The reason why we selected a simpler path model comes from the need to model initially TCP, or at least some parts of its behavior, so that we can use these results online by our system. By following a more complex path model, and therefore TCP model, the overall system complexity will be unnecessarily increased. With the Bernoulli model the only quantity needed to model the channel is the average packet loss rate. The average packet loss rate is simply calculated by dividing the number of lost packets with the total number of packets lost or received:

$$P_{\mathrm{L}} = \frac{\#\,\text{lost}}{\#(\text{lost} + \text{received})}. \tag{1}$$

4. Packet-level analysis of video streaming with TCP

Our primary goal is to understand on the packet level, how TCP would send a stream of video packets that are packetized as groups of encoded macroblocks. Essentially, we want to be able to identify precisely which macroblocks are expected to be lost when the TCP packet that contains them is lost. Ultimately this knowledge will be used for estimating the expected distortion at the decoder. An important point that must be clear before we start our analysis is that we are concerned only with the congestion avoidance phase and not the initial slow start. We think that this assumption is valid in a practical setup, since a streaming session represents a persistent data transfer that will spent most of the TCP connection lifetime in the congestion avoidance phase. During this period we want to understand TCP behavior, and act appropriately at the encoder.

Let us revisit Fig. 3 and see in more detail the adopted packetization method for our TCP streaming system. We assume of course a constant number of macroblocks per frame m, while the number of encoded macroblocks that can fit into a TCP packet is q. According to our streaming model, the encoder and packetizer commit RTP packets (header plus encoded macroblocks) to TCP. The encoded macroblocks that will be contained in a single TCP packet will probably belong to more than one frame, if the frame size is smaller than the allowed TCP payload.[1] There are two values that we want to know and these are the actual lost macroblocks $\mathbf{M}_{\text{lost}}$,[2] and the projected lost macroblocks $\tilde{\mathbf{M}}_{\text{lost}}$. This first value is the result of actual measurements since the TCP sender has knowledge of the specific packets that were lost. This is information is used at the encoder so that $\tilde{\mathbf{M}}_{\text{lost}}$ and the distortion estimate can be appropriately updated. However, the estimation of the expected number of lost macroblocks depends on the adopted channel model and will be our next topic.

4.1. Loss probabilities for TCP packets and macroblocks

Estimating which TCP packets are expected to be lost is not a trivial task. Given the nature of Internet routers which might drop packets at their discretion when buffer overflows occur, we cannot project, even in the short term, the fate of the packets that we will sent. However, stochastic modeling of the Internet, and more specifically of TCP, has given tools with which we can estimate the long-term protocol behavior [23,2]. These models assume a known channel behavior, based on which they make estimates regarding the evolution of TCP congestion window.

Have a look at Fig. 4, where we demonstrate how the TCP congestion window is evolved until a packet loss is identified at the sender based on a triple-duplicate (TD) event. Each vertical stack of blocks denotes the number of packets sent during a time frame of one RTT. The time frame between two successive loss events is called a no-loss round (NLR). After a packet loss, and at the start of the new NLR j, the TCP congestion window will grow

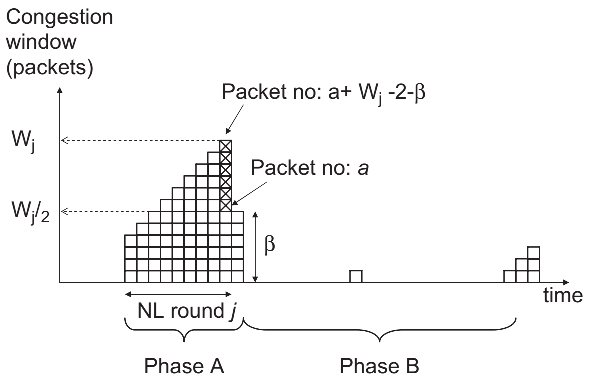

Fig. 4. Congestion window evolution for TCP.

from W_{j-1} packets to W_j. If we assume that the TCP packet with sequence number a is lost, all the subsequent packets in the same RTT round will also be lost [1], since we assumed a Bernoulli packet loss model. But because $a-1$ packets have been acknowledged, then W_j-1 more packets will be sent. From these new packets β_j will be received correctly, since they are sent in the next RTT round as seen in Fig. 4. Therefore, in total they will be lost: $l_j = 2W_j - a_j - \beta_j - 1$ packets. For simplicity we assume that $\beta_j = W_j/2$, which is a good approximation. Also, since we adopt the Bernoulli packet loss model, the congestion window will be evolved according to the following equation [23]:

$$W(P_{\text{L}}) = \frac{2}{3} + \sqrt{\frac{4(1-P_{\text{L}})}{3P_{\text{L}}} + \frac{4}{9}}. \tag{2}$$

From this simple analysis we can understand that essentially we would like to know what is the value of a_j, or else the expected number of correctly sent packets during an NLR j in TCP congestion avoidance phase. But the value of a_j depends on the adopted channel model, which in our case follows a known Bernoulli distribution. Therefore, if the average packet loss rate is p, the probability that $k-1$ are received before a loss occurs will be $P[a=k] = (1-p)^{k-1}p$. Therefore, the average value of a will be $1/p$. In order to exploit this result, our streaming system measures the average packet loss rate p as we described in Section 3.2. Based on this value it estimates the expected number of sent/acknowledged packets during an NLR $\tilde{a}$, and also checks the actual number of sent/acknowledged packets. Intuitively, The more packets we have sent without a loss, the highest the probability that loss

[1]This value depends on the maximum segment size. For QCIF frames typical sizes of P frame macroblocks are in the area of 100 bytes.

[2]Bold letters indicate vectors.

will take place close to $\tilde{a}$, and the congestion window will be halved.

It is also possible that a packet loss is identified not with TD event but with a timeout (TO). If this happens then TCP would additionally decrease the congestion window to one packet leading to a halt in the data flow. Our analysis in that situation would not be different since our goal is to estimate $\tilde{a}$. The fact that the congestion window will be drastically reduced (one packet), and will therefore contribute to excessive delay for subsequent packets, will be analyzed in the next subsection.

4.1.1. Complete frames within an NLR

Our goal now is to find the macroblocks that correspond to the TCP packets that are expected to be lost with the probability that we previously calculated. In order to start our analysis, assume that in each NLR only complete frames are sent starting from the first macroblock of a frame. Later we will relax this assumption so that macroblocks of a frame can be transmitted in the next NLR. Recall that we assume a constant number of macroblocks per frame m, and the number of encoded macroblocks that can fit into a TCP packet is q. Based on the packetization method explained in the previous section, the total number of correctly sent macroblocks will be aq, and the total number of lost macroblocks will be lq, making thus the number of complete frames sent equal to $\lfloor aq/m \rfloor$. Therefore, the sequence of complete frames that are expected to be sent in an NLR only are the ones with id:

$$\tilde{\mathbf{F}}_{\text{complete}} = \left\{0, \ldots, \left\lfloor \frac{aq}{m} \right\rfloor - 1\right\}. \tag{3}$$

This makes the number of MBs that were sent and lost equal to $aq - m\lfloor aq/m \rfloor$. In addition we know that the TCP packets with sequence number a till $a + W_j + \beta - 2$ will be retransmitted meaning that the MBs with id aq till $lq = (a + W_j + \beta - 2)q$ will be retransmitted. Let the notation M_i^n denote the macroblock with id i that belongs to frame n. Then the estimate is that the following vector of macroblocks will be lost:

$$\tilde{\mathbf{M}}_{\text{lost}}^{j} = \{M_{aq}^{n}, \ldots, M_{(a+W_j+\beta-2)q}^{n+\lfloor aq/m \rfloor -1}\} \tag{4}$$

and also the following macroblock sequence will be received correctly:

$$\tilde{\mathbf{M}}_{\text{correct}}^{j} = \{M_{0}^{n}, \ldots, M_{aq-1}^{n+\lfloor aq/m \rfloor -1}\}. \tag{5}$$

The additional latency incurred by the lost macroblocks will be calculated in the next section. This fact must be taken into account when calculating the distortion estimate, since the probability of arriving on time will be reduced.

4.1.2. Frames separated across NLRs

Another case that has to be considered is when at the start of new NLRs, the sender must transmit remaining macroblocks that belong to a previous frame. Assume that at the start of the NLR j a number of h macroblocks from frame n have to be transmitted, through the creation of new slice. Then we have for the sequence of lost macroblocks

$$\tilde{\mathbf{M}}_{\text{lost}}^{j} = \{M_{m-h}^{n}, \ldots, M_{m-1}^{n}, \ldots, M_{(a+W_j+\beta-2)q}^{n+\lfloor aq/m \rfloor -1}\}. \tag{6}$$

The previous equation is important since we know the vector of macroblocks that were lost, allowing us thus to quantify their effect on the expected distortion.

Since we said that the path is assumed to generate packet loss according to the Bernoulli model with average packet loss rate p, the probability that a packets are received is

$$P_a = (1 - p)^a p. \tag{7}$$

If we denote as η_i^n the TCP packet that contains macroblock i of frame n, the expected macroblock loss probability $P_{\text{L}}^{(n,i)}$ is

$$P_{\text{L}}^{(n,i)} = \begin{cases} 0 & \text{if } n = 0, \\ 1 - (1-p)^{\eta_i^n} & \text{if } n > 0, \end{cases} \tag{8}$$

and of course $P_{\text{R}}^{(n,i)} = 1 - P_{\text{L}}^{(n,i)}$. The usefulness of Eq. (6) is to estimate which macroblocks belong to the same TCP packet and therefore have the same loss probability as calculated by Eq. (8).

4.2. Modeling the packetization method

The next step in our methodology is to formalize the packetization methods that in our case can be macroblock row interleaving and scattered slices. Our goal with this step is to quantify in the expected distortion the effect of the selected packetization method. Online change of the selected packetization method might not be useful, but as it has been observed sequences with different spatial and temporal patterns exhibit different PSNR quality according to the selected duo packetization/EC methods [28]. Essentially, the nature of the encoded video affects the efficiency of the selected error-concealment method. However, we do not delve into this issue in this paper.

Having said that, someone can easily see that the distinction of different packetization methods resides in the selection of the packet in which macroblock M_i^n will be placed. Consider the method of scattered slices, where adjacent macroblocks in scan order are placed in different slices. This means that with q macroblocks per TCP packet, M_i^n would be placed in TCP packet (from Fig. 2)

$$\eta_i^n = 2\left\lfloor \frac{i}{2q} \right\rfloor + (i \bmod 2). \tag{9}$$

For the macroblock row interleaving case, each line of macroblocks corresponds to a single slice. Therefore, the form would be simpler:

$$\eta_i^n = \left\lfloor \frac{i}{q} \right\rfloor. \tag{10}$$

The above two equations can be substituted in to Eq. (8), in order to obtain the new macroblock loss probabilities that represent different packetization methods.

5. TCP packet latency

What we did until this point is to identify the lost TCP packets and the corresponding macroblocks. However, our job is not done since we have to calculate the expected latency for the TCP packets, which might have been lost or not, since excessive latency may result in useless video packets at the decoder. If t_s and t_d represent the time sent and the deadline for M_i^n, respectively, then we denote the maximum allowed latency as $\Delta t^{(n,i)} = t_d^{(n,i)} - t_s^{(n,i)}$. Clearly, Δt might have a negative value since the deadline might be missed even before we send the packet. However, this case does not affect our analysis. With TCP, a lost packet is identified either when three duplicate acknowledgments (TD) arrive at the sender leading to an immediate fast retransmission, or when a TO event takes place [27]. Therefore, let P_{TD} and P_{TO} denote the probability of a TD event that leads to a fast retransmission, or an RTO expiration, respectively. Then we can write for the probability of macroblock M_i^n to be late based on the three possible events (received, TD, TO):

$$\begin{aligned} P_{\mathrm{D}}^{(n,i)} &= P\{L \geqslant \Delta t\} = 1 - P\{L < \Delta t\} \\ &= 1 - P\{L_{\mathrm{N}} < \Delta t^{(n,i)}\} P_{\mathrm{R}}^{(n,i)} \\ &\quad - P\{L_{\mathrm{TD}} < \Delta t^{(n,i)}\} P_{\mathrm{TD}} P_{\mathrm{L}}^{(n,i)} \\ &\quad - P\{L_{\mathrm{TO}} < \Delta t^{(n,i)}\} P_{\mathrm{TO}} P_{\mathrm{L}}^{(n,i)}. \end{aligned} \tag{11}$$

The above equation is crucial since we want to quantify these events, and therefore our goal in this section is to calculate this probability. The problem now reduces to finding the distribution of the end-to-end network latency L_{N}, the fast retransmission latency L_{TD}, and the latency from an RTO expiration L_{TO}. Another task is to calculate P_{TD} and P_{TO}.

The important point is that the latency L_{N} refers only to the network and not delays related to the TCP retransmission mechanisms. Therefore, the selection of $f_{L_{\mathrm{N}}}$ is a decision orthogonal to our work in this paper. In our case we will model the one-way Internet latency as a gamma distribution that occurs mainly due to buffering at the routing infrastructure [21]:

$$f_{L_{\mathrm{N}}}(t) = \begin{cases} \dfrac{\lambda \mathrm{e}^{-\lambda t}(\lambda t)^{v-1}}{(v-1)!} & \text{if } t \geqslant 0, \\ 0 & \text{if } t < 0. \end{cases} \tag{12}$$

Several possible analyses can be performed in order to model more accurate the core network performance, but this research is out of the scope of this paper.

5.1. Probability of TD and TO events

In order to avoid duplicate work, we will use from the well-known TCP model [23], the probabilities for a packet loss to be a TO (P_{TO}) or TD (P_{TD}) events. For a Bernoulli path with an average end-to-end packet loss probability p, the probability that a packet loss is a TO is

$$P_{\mathrm{TO}} = \min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right),$$

and of course when the packet loss is not a TO it is a TD making thus $P_{\mathrm{TD}} = 1 - P_{\mathrm{TO}}$.

5.2. Retransmitted packet latency

Regarding packets identified with a TD event, we assume that the retransmission will arrive at the receiver. These packets will incur an additional latency of one RTT since one backward trip time must be consumed for the duplicate acknowledgments to arrive at the sender, and one forward trip for the retransmission to arrive at the receiver.

Concerning the packets recovered with TO, a more elaborate analysis is needed. Recall again Fig. 4 and more specifically what we call phase B in this figure. This case simply considers the behavior of the congestion window which is one segment, when a TO has taken place. In the previous subsection we obtained a value for the probability of a TO happening. Therefore, we want to know what will be the expected duration of a sequence of TO events.

For the jth NLR the expected RTO duration will be [23]

$$\tilde{L}^j_{\mathrm{TO}} = \mathrm{RTO}_0\,\frac{1+p^2+2p^2+4p^3+8p^4}{1-p}, \tag{13}$$

while the actual duration will be L^j_{TO}.

6. Analytical model of the expected decoder distortion

Based on the macroblock and packet loss probabilities that we calculated, our goal in this section is to derive analytically the expected distortion at the decoder as a function of the latency introduced by TCP, the packetization method, and the error-concealment method used at the decoder.

One of the assumptions that we make is that the first intraframe in the sequence is always received, even after a number of retransmissions must take place. We have named M^n_i the coded macroblock at location i of frame n. Let also f denote the pixel value at the encoder, $\tilde{f}$ the reconstructed value at the decoder, and $\hat{f}$ the encoder estimation of the reconstructed pixel value at the decoder. We denoted as η^n_i the TCP packet used to packetize M^n_i, and as K the number of packets that packetize the first I frame of the series. We also calculated in the previous section the standalone probabilities for macroblocks to be delayed $P^{(n,i)}_{\mathrm{D}}$, or received in time $P_{\mathrm{R}} = (1-P_{\mathrm{D}})(1-P_{\mathrm{L}})$.

In addition, we want to model the two simple error-concealment methods that we described in Section 3. Therefore, we have to take into consideration the effect of delayed packets that can be used for concealment. For EC1, we have to calculate what is the probability for M^n_i to be delayed, given that the macroblock which can conceal it (M^{n-1}_i) was received in time ($P^{(n,i)}_{\mathrm{RD}}$), or it was also delayed ($P^{(n,i)}_{\mathrm{DD}}$). For spatio-temporal error-concealment mode EC2, we have to calculate the same quantities but for macroblock M^n_{i-1}. We also made the assumption that the first frame $n=0$ is always received, which means that $P^{(n,i)}_{\mathrm{RD}}$ and $P^{(n,i)}_{\mathrm{DD}}$ will be equal to zero. In addition, when M^n_i and M^{n-1}_i are placed in the same packet, this means that the macroblock which is valid for concealment has been received and with it the macroblock that could conceal. Therefore in this case the probability $P^{(n,i)}_{\mathrm{RD}}$ will be zero. Therefore, $P^{(n,i)}_{\mathrm{RD}}$ is written as

$$P^{(n,i)}_{\mathrm{RD}} = \begin{cases} 0 & \text{if } n=0,\\ P^{(n-1,i)}_{\mathrm{R}}P^{(n,i)}_{\mathrm{D}} & \text{if } n>0,\ \text{EC1 and } \eta^n_i \neq \eta^{n-1}_i,\\ 0 & \text{if } n>0,\ \text{EC1 and } \eta^n_i = \eta^{n-1}_i,\\ P^{(n,i-1)}_{\mathrm{R}}P^{(n,i)}_{\mathrm{D}} & \text{if } n>0,\ \text{EC2 and } \eta^n_i \neq \eta^{n}_{i-1},\\ 0 & \text{if } n>0,\ \text{EC2 and } \eta^n_i = \eta^{n}_{i-1}. \end{cases} \tag{14}$$

The non-zero terms in the above equation are explained as follows: the probability for M^n_i to be delayed, precisely after the successful delivery of the required MB for concealment, is given as the multiplication of $P^{(n-1,i)}_{\mathrm{R}}P^{(n,i)}_{\mathrm{D}}$ for EC1 and $P^{(n,i-1)}_{\mathrm{R}}P^{(n,i)}_{\mathrm{D}}$ for EC2, when the two events are independent (i.e. they belong to different packets).

Similar for the probability that M^n_i was delayed, given that the macroblock which can conceal it (M^{n-1}_i) was also delayed:

$$P^{(n,i)}_{\mathrm{DD}} = \begin{cases} 0 & \text{if } n=0,\\ P^{(n-1,i)}_{\mathrm{D}}P^{(n,i)}_{\mathrm{D}} & \text{if } n>0,\ \text{EC1 and } \eta^n_i \neq \eta^{n-1}_i,\\ 1 & \text{if } n>0,\ \text{EC1 and } \eta^n_i = \eta^{n-1}_i,\\ P^{(n,i-1)}_{\mathrm{D}}P^{(n,i)}_{\mathrm{D}} & \text{if } n>0,\ \text{EC2 and } \eta^n_i \neq \eta^{n}_{i-1},\\ 1 & \text{if } n>0,\ \text{EC2 and } \eta^n_i = \eta^{n}_{i-1}. \end{cases} \tag{15}$$

The next step is the calculation of the expected value of a reconstructed pixel at the decoder. Eqs. (16) and (17) depict the expected pixel values $\hat{f}^n_{x,y}$, for intracoded and intercoded macroblocks. The pixel value will be equal to the reconstructed value at the encoder times the probability to receive the MB correctly $P^{(i,n)}_{\mathrm{R}}\tilde{f}^n_{x,y}$, plus the probability to lose this MB, and so use the reconstructed value of the same pixel of the previous frame $\hat{f}^{n-1}_{x,y}$ (error-concealment is used). In the second case, if there is an MB before it, the decoder uses a different EC method. This event requires the addition of the probability that this MB is lost and the previous

MB (M_i^{n-1}) was received correctly which will be the value of this pixel in the previous frame ($\hat{f}_{u,v}^{n-1}$). Finally, we have to add the probability that both the previous and the current MB are lost and another pixel from the previous frame ($\hat{f}_{x,y}^{n-1}$) is used. The value $\tilde{e}_{x,y}^{n}$ in the following equations denotes the IDCT residue that is added to the reconstructed pixel values:

$$\hat{f}_{x,y}^{n} = \begin{cases} P_{\mathrm{R}}^{(n,i)}\tilde{f}_{x,y}^{n} + P_{\mathrm{RD}}^{(n,i)}\hat{f}_{x,y}^{n-1} \\ \quad + P_{\mathrm{DD}}^{(n,i)}\hat{f}_{x-1,y}^{n} & \text{if EC1/EC2,} \\ P_{\mathrm{R}}^{(n,i)}\tilde{f}_{x,y}^{n} + P_{\mathrm{RD}}^{(n,i)}\hat{f}_{x-1,y}^{n} \\ \quad + P_{\mathrm{DD}}^{(n,i)}\hat{f}_{x,y}^{n-1} & \text{if EC2/EC1,} \end{cases} \tag{16}$$

$$\hat{f}_{x,y}^{n} = \begin{cases} P_{\mathrm{R}}^{(n,i)}(\tilde{e}_{x,y}^{n} + \hat{f}_{u,v}^{n-1}) + P_{\mathrm{RD}}^{(n,i)}\hat{f}_{x,y}^{n-1} \\ \quad + P_{\mathrm{DD}}^{(n,i)}\hat{f}_{x-1,y}^{n} & \text{if EC1/EC2,} \\ P_{\mathrm{R}}^{(n,i)}(\tilde{e}_{x,y}^{n} + \hat{f}_{u,v}^{n-1}) + P_{\mathrm{RD}}^{(n,i)}\hat{f}_{x-1,y}^{n} \\ \quad + P_{\mathrm{DD}}^{(n,i)}\hat{f}_{x,y}^{n-1} & \text{if EC2/EC1.} \end{cases} \tag{17}$$

After calculating the expected pixel values, the global distortion will be given as the mean absolute differences (MAD) of the pixel values in frame n, for the either intra- or intercoded MB i:

$$\mathrm{MAD}(M_i^n) = \frac{\sum_{y=1}^{16}\sum_{x=1}^{16}|f_{x,y}^{n,i} - \hat{f}_{x,y}^{n,i}|}{256}. \tag{18}$$

7. Encoding mode selection with Lagrangian optimization

The question that rises now is how to utilize this analytical model of the expected decoder distortion. What we claim is that the more accurate distortion estimate at the encoder can lead to a better allocation of the available TCP rate through selection of the encoding mode of each individual macroblock. Even though this principle has been demonstrated before [36,33], in this paper we are the first to consider the effect of a transport protocol, and the specific loss pattern that introduces, in the behavior of the real-time encoder.

The next question that has to be answered is how to select the encoding mode for each macroblock. This decision must be done so that the encoded macroblocks conform to the bit budget calculated by the encoder rate control algorithm. Our task is to allocate the bit budget per frame to each of the macroblocks that are about to be encoded, by selecting an intra- or predictive encoding mode. To formalize this problem, consider the group of m macroblocks that belong to frame n, i.e. $\mathbf{M}^n = (M_1^n, \ldots, M_m^n)$. Also consider the encoding vector for these macroblocks $\mathbf{\Theta}^n = (\theta_1^n, \ldots, \theta_m^n)$, where $\theta \in \{\text{intra}, \text{inter}\}$. If there is a number of N frames waiting to be encoded, with $\mathbf{\Phi} = (\mathbf{M}^1, \ldots, \mathbf{M}^N)$, the objective is to

$$\min E[D(\mathbf{\Phi}, \mathbf{\Theta})] \quad \text{such that } R(\mathbf{\Phi}) \leqslant R_{\mathrm{c}}. \tag{19}$$

R_{c} is the current rate constraint imposed by TCP. The expected distortion for a macroblock $i \in (1, \ldots, m)$ that belongs to frame n will be $D[M_i^n] = \mathrm{MAD}(M_i^n)$, that was derived in Eq. (18). Now, the objective of the Lagrangian relaxation problem we define is to select the optimal vector $\mathbf{\Theta}^*$ so that the overall distortion is minimized.

The above constrained Lagrangian optimization problem can be formally defined in an unconstraint form for all N frames up to be encoded:

$$\min \sum_{s=1}^{mN} J_s = \min\left(\sum_{s=1}^{mN} E[D(\mathbf{\Phi}, \mathbf{\Theta})] + \lambda \sum_{s=1}^{mN} R(\mathbf{\Phi})\right). \tag{20}$$

The selection of the optimal Lagrange multiplier λ can be selected using several alternatives. We followed the same approach as in [34] in order to simplify comparison. So for a frame n this parameter is set as

$$\lambda_n = \frac{2B_n + (g - B_n)}{B_n + (g - B_n)}\lambda_{n-1}, \tag{21}$$

where B_n denotes the current occupancy of the encoder buffer before frame n was encoded.

8. Implementation issues

Based on our analysis that we performed, our objective now is to define a practical streaming protocol. In Fig. 5, we present in pseudo-code the final protocol, which is based on the previous analysis and derived analytical results. As it can be understood, identifying the optimal encoding vector $\mathbf{\Theta}^*$ is a time consuming task since the MAD operation must be performed twice for the m macroblocks of a single frame. For the QCIF sequences (176×144), we have that $m = 99$, and therefore 198 MAD calculations are performed and

```
tcp_streaming_sender()
1: accept TCP connection
2: for sending NLR $j$ do
3:   calculate $\tilde{L}^j_{TO}$, $\tilde{L}^j_{TD}$, $\tilde{L}^j_N$
4:   calculate $\tilde{P}^j_{TO}$, $\tilde{P}^j_{TD}$
5:   calculate $a_j$
6:   for frame $n$ ready for encoding do
7:     est. $\tilde{\mathbf{M}}^n_{lost}$ // based on the assign. to TCP pkts
8:     for all MBs $\in n$ (This is equal to $m$) do
9:       calculate $\tilde{P}^{(j,n,i)}_D$
10:      calculate twice $MAD(M^{(n,i)})$ for intra/inter
11:    end for
12:    for all $i \in (1, ..., m)$ (all MBs) do
13:      $J^{intra}_i = D(M^{(n,i)}_{intra}) + \lambda_n R(M^{(n,i)}_{intra})$
14:      $J^{inter}_i = D(M^{(n,i)}_{inter}) + \lambda_n R(M^{(n,i)}_{inter})$
15:      $\theta^*_i = min(J^{intra}_i, J^{inter}_i)$
16:    end for
17:    create $\mathbf{\Theta}^*$
18:  end for
19: end for
tcp_streaming_client()
1: setup TCP connection
2: send clip name, EC mode, packetization
3: send RTCP reports (actual $M_{lost}$ and $M_{delayed}$ if needed by the server)
```

Fig. 5. Protocol for video streaming with TCP.

the minimum for each individual macroblock is selected.

The implementation of real-time encoding is a challenging task with respect to the performance of the actual software implementation. Especially with new codecs like H.264, the complexity of the encoder requires significant processing costs. However, the reader can understand that our methodology is not tied to the specific codec that we are using. In fact any of the codecs that use the hybrid spatio/temporal principle for encoding can be used. We selected H.263 primarily for its use in interactive video communication applications and for its simple and fast software implementation [11].

9. Experiments

The network setup shown in Fig. 6 was used throughout our experiments in this paper. The scenario assumes a sender and a receiver that are

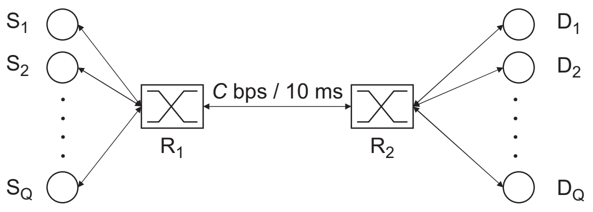

Fig. 6. Experimental setup.

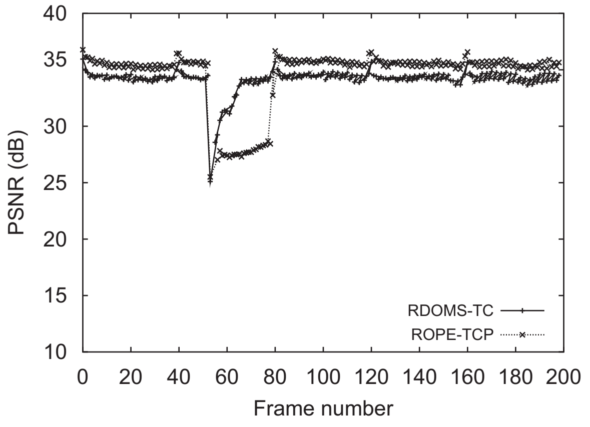

Fig. 7. PSNR as a function of the frame number at the encoder. Sequence MISS AMERICA. Target encoding rate is 512 kbps and bottleneck $C = 512$ kbps.

linux boxes while a freeBSD machine is used for controlling the bottleneck link. The Dummynet software [7] was used in the middlebox in order to emulate various link configurations in terms of packet loss rate, bandwidth, and delay. The QCIF FOREMAN, CARPHONE, and MISS AMERICA sequences [29] were used for real-time encoding with the $H.263++$ encoder [11] at various bitrates. Slices that consist of 11 macroblocks were packetized into RTP packets and then sent to TCP (nine slices per frame). Due to the short duration of the sequences (150 frames), they were repeatedly fed as input to the encoder. The capacity of the bottleneck link between the two routers is set at C bps while the delay was constant at 10 ms. The results were obtained by running the same scenario 100 times and averaging the PSNR values of the same experiments.

Encoder dynamics: We start presenting our experimental results by showing in Fig. 7 the PSNR at the encoder as a function of the frame number. For our comparative experiments we implemented the ROPE algorithm [36] while TCP was also used for transport. The first observation is that the ROPE rate–distortion metric provides a more generous encoding rate leading to a slightly higher PSNR when no packet loss takes place. In the case of RDOMS-TCP, the metric is slightly more pessimistic about the state of transport channel/protocol. While frame number 60 was being encoded, the Bernoulli path model generated a burst packet loss at the bottleneck, leading to a fast retransmission of several packets by TCP. When this happened, for RDOMS-TCP the short-term latency is expected to be increased sharply for the retransmitted packets (even if no further packet loss is observed). This means that retransmitted macroblocks will be late, since we use EC1 and at the decoder they are being replaced by the same macroblock of the previous frame. The RDOMS-TCP MAD estimate in this case will be more precise (lower value), while the ROPE-TCP MAD will have higher value since it ignores this retransmission. This allows more bits to be freed for encoding of other macroblocks as intra, increasing thus encoder PSNR. However, with ROPE-TCP the algorithm considers the bad channel state but for few dropped packets, and assumes that when the bad state is over the protocol can fully exploit the channel.

Distortion estimate: The above analysis helped us highlight the operating flow of our algorithm in the real-time encoder. The real objective is to demonstrate that the more precise distortion estimates, based on the additional protocol parameters, can lead to significant quality improvements at the decoder. Therefore we will now evaluate the accuracy of the distortion metric. However, the ROPE algorithm was developed by assuming a two-state Gillbert–Elliot underlying channel model, without accounting, however, for the possible use of TCP. This configuration might seem unfair, but recall that our objective is to evaluate the performance of our new distortion model which accounts for TCP (TCP-RDM).

Fig. 8 presents comparative results of the actual distortion, for the ROPE and TCP-RDM metrics for the MISS AMERICA sequence for an end-to-end channel packet loss probability of 10^{-3}. What we see in this figure is that ROPE overshoots the actual decoder PSNR. This is of course something to be expected since the algorithm was not engineered in the first place to consider TCP. However, the results for the proposed TCP-RDM highlight essentially our main idea that a more accurate distortion estimate can be obtained if we consider TCP when it is used for video streaming. Results for the same experiment but for the sequence CARPHONE are shown in Fig. 8(b), and we observe that same pattern happens there too but with higher mismatches. The reason is that the CARPHONE sequence is characterized by more sudden movement pattern between successive video frames.

In Fig. 9 we present a different set of results that were obtained when we increased the end-to-end channel packet loss probability to 10^{-2}. By setting the packet loss probability at a higher value we wanted essentially to test the behavior of the model since TCP will suffer from an increased number of TD events and TO. The distortion estimate

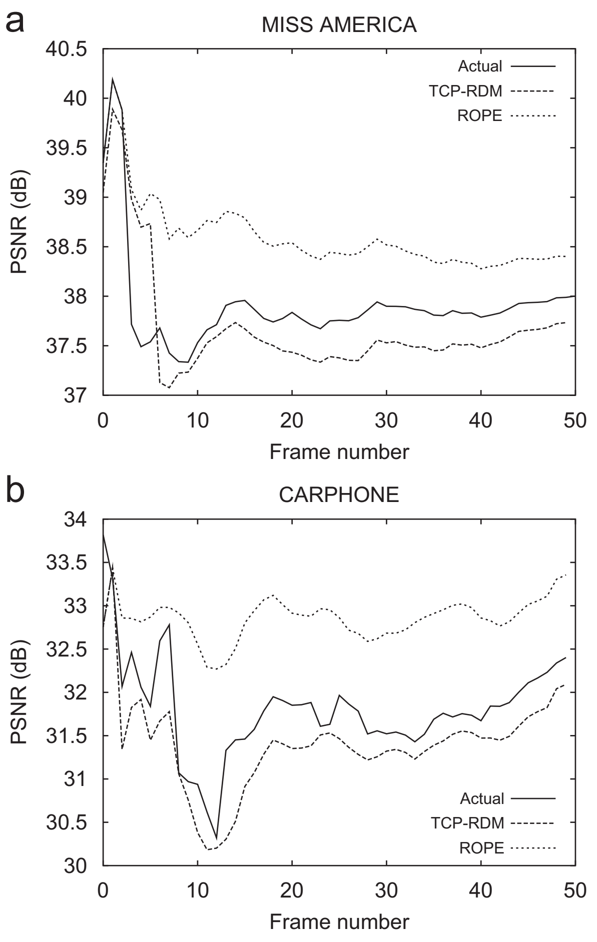

Fig. 8. Comparative results of the distortion estimates and the decoder PSNR. Target encoding rate is 256 kbps and bottleneck $C = 512$ kbps. Packet loss probability: 10^{-3}. (a) MISS AMERICA and (b) CARPHONE.

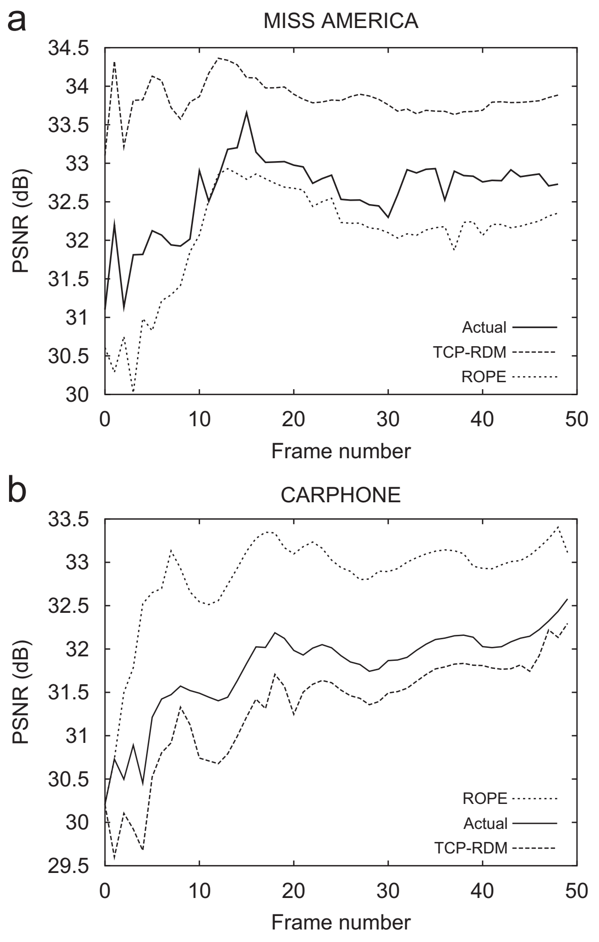

Fig. 9. Comparative results of the distortion estimates and the decoder PSNR for increased packet loss. Target encoding rate is 256 kbps and bottleneck $C = 512$ kbps. Packet loss probability: 10^{-2}. (a) MISS AMERICA and (b) CARPHONE.

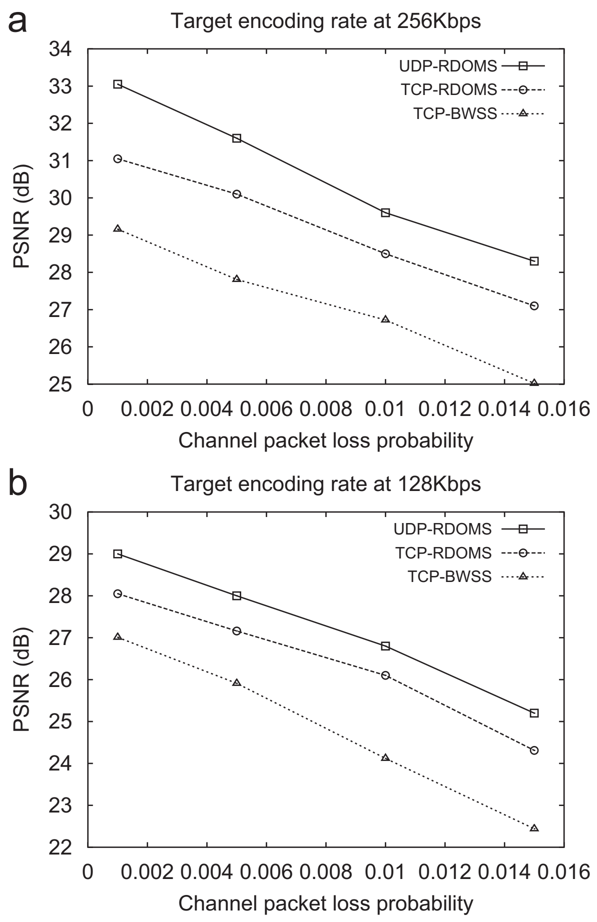

Fig. 10. PSNR as a function of the end-to-end packet loss probability for video streaming with TCP and the sequence FOREMAN. (a) Target encoding rate at 256 kbps and (b) target encoding rate at 128 kbps.

with TCP-RDM is even better from the previous experiments as it can be seen in Fig. 9 for both MISS AMERICA and CARPHONE sequences. What happens is that the channel is not utilized properly since TCP TO last even after the channel has been returned into the good state.

Encoding mode selection: The next step is to evaluate the performance of the proposed encoding mode selection algorithm. For our comparison in this section, we implemented the BWSS algorithm [19] which is a methodology that attempts to minimize TCP bandwidth fluctuations. This last method is a purely transport level optimization that does not involve any distortion estimates. However, we think that it is necessary to compare this algorithm with our work in order to demonstrate the effectiveness of different approaches.

Fig. 10 presents PSNR as a function of the channel packet loss probability for real-time target encoding rate of 256 kbps and 64 kbps, respectively. We compare our approach with reproduced results from [33] that implements an RD optimal mode selection (RDOMS) policy for streaming with UDP, and the approach at [19] that also considers streaming with TCP. We see that when the target bitrate was 256 kbps, the RDOMS/UDP approach outperforms both the other two. However, the benefit of the proposed RDOMS algorithm comes into place when TCP is used for transport. It clearly outperforms by 2–2.5 dB, a purely TCP-based streaming approach, which does not consider the protocol behavior. More importantly, for higher packet loss rate, the performance in terms of PSNR is increasing. When the target bitrate was set to

64 kbps, PSNR presents the same trend, but this time the effect is not so severe, due to the lower bitrate injected to the network.

10. Conclusions

In this paper we presented a mechanism for video streaming with the transmission control protocol (TCP) that uses a new RD metric (TCP-RDM) that characterizes the expected video distortion at the decoder. Initially, we developed this metric so that it considers TCP introduced latency, the packetization method, and error-concealment method at the receiver. Based on this analytical distortion model we proposed an algorithm for rate–distortion optimized mode selection (RDOMS-TCP). This algorithm specifies the encoding of each macroblock in intra- or predictive mode. Experimental results for real-time encoded video streaming showed PSNR improvement of nearly 2 dB over other methods that we tested for TCP video streaming.

Our goal was to demonstrate that TCP presents a viable solution for the transport of real-time encoded video bitstreams. We think that our work is one step towards this direction, since we showed that if additional optimizations are applied at the encoder, further quality improvement can be observed. We believe that the wide-scale deployment of TCP, and the implementation of the proposed algorithm at the application level, can lead to a practical system.

References

[1] A.A. Abouzeid, S. Roy, M. Azizoglu, Stochastic modeling of TCP over lossy links, in: INFOCOM, 2000.

[2] N. Cardwell, et al., Modeling TCP latency, in: INFOCOM, 2000.

[3] J. Chakareski, B. Girod, Rate–distortion optimized packet scheduling and routing for media streaming with path diversity, in: IEEE Data Compression Conference (DCC), 2003.

[4] P.A. Chou, Z. Miao, Rate–distortion optimized streaming of packetized media, Microsoft Research Technical Report MSR-TR-2001-35, 2001.

[5] P.A. Chou, A. Sehgal, Rate–distortion optimized receiver-driven streaming over best-effort networks, in: IEEE International Packet Video Workshop, 2002.

[6] G. Cote, S. Shirani, F. Kossentini, Optimal mode selection and synchronization for robust video communications over error prone networks, IEEE J on Sel. Areas in Comm. 18 (6) (June 2000) 952–965.

[7] Dummynet. ⟨http://www.info.iet.unipi.it/luigi/ip_dummynet/⟩.

[8] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Trans. on Networking 7 (4) (August 1999) 458–472.

[9] P. Frossard, O. Verscheure, Joint source/FEC rate selection for quality optimal MPEG-2 video delivery, IEEE Trans. on Image Process. 10 (12) (December 2001) 1815–1825.

[10] E. Gurses, G.B. Akar, N. Akar, A simple and effective mechanism for stored video streaming with TCP transport and server-side adaptive frame discard, Comput. Networks 48 (4) (July 2005) 489–501.

[11] H.263 codec. ⟨http://www.xs4all.nl/roalt/h263.html⟩.

[12] M. Handley, S. Floyd, J. Pahdye, J. Widmer, TCP friendly rate control (TFRC): protocol specification, RFC 3448, January 2003.

[13] P.-H. Hsiao, H.T. Kung, K.-S. Tan, Video over TCP with receiver-based delay control, in: ACM NOSSDAV, 2001.

[14] V. Jacobson, Congestion avoidance and control, in: ACM SIGCOMM, August 1988, pp. 314–329.

[15] M. Kalman, B. Girod, Techniques for improved rate–distortion optimized video streaming, ST J. of Res. 2 (1) (November 2005) 45–54.

[16] C. Krasic, K. Li, J. Walpole, The case for streaming multimedia with TCP, in: Workshop on Interactive Distributed Multimedia Sytems (IDMS), 2001.

[17] S. Liang, D. Cheriton, TCP-RTM: using TCP for real time applications, in: ICNP, 2002.

[18] Y.J. Liang, B. Girod, Prescient RD optimized packet dependency management for low-latency video streaming, in: IEEE International Conference on Image Processing (ICIP), vol. 2, September 2003, pp. 659–662.

[19] P. Mehra, A. Zakhor, TCP-based video streaming using receiver-driven bandwidth sharing, in: International Packet Video Workshop, 2003.

[20] Z. Miao, A. Ortega, Expected runtime distortion based scheduling for delivery of scalable media, in: International Conference of Packet Video, 2002.

[21] A. Mukherjee, On the dynamics and significance of low frequency components of Internet load, Technical Report, University of Pennsylvania, Technical Report MS-CIS-92-83, 1992.

[22] B. Mukherjee, T. Brecht, Time-lined TCP for the TCP-friendly delivery of streaming media, in: ICNP, 2000.

[23] J. Padhye, V. Firoiu, D. Towsley, Modeling TCP reno performance: a simple model and its empirical validation, IEEE/ACM Trans. on Networking 8 (2) (April 2000) 133–145.

[24] Quicktime. ⟨http://www.apple.com⟩.

[25] Real media, ⟨http://www.realnetworks.com⟩.

[26] J. Rosenberg, L. Qiu, H. Schulzrinne, Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet, in: INFOCOM, 2000.

[27] Richard Stevens, TCP/IP Illustrated Volume 1, Addison-Wesley, Reading, MA, 1994.

[28] M.-T. Sun, A.R. Reibman, Compressed Video Over Networks, Marcel Dekker, New York, September 2001.

[29] TML video sequences, Available from: ⟨http://www.kbs.cs.tu-berlin.de/stewe/vceg/sequences.htm⟩.

[30] B. Wang, J. Kurose, P. Shenoy, D. Towsley, Multimedia streaming via TCP: an analytic performance study, in: ACM Multimedia, 2004.

[31] Y. Wang, J. Ostermann, Y.-Q. Zhang, Video Processing and Communications, Prentice-Hall, Englewood Cliffs, NJ, 2002.

[32] Windows media, ⟨http://www.microsoft.com⟩.
[33] D. Wu, T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang, H.J. Chao, On end-to-end architecture for transporting MPEG-4 video over the Internet, IEEE Trans. on Circuits and Systems for Video Technol. 10 (6) (September 2000) 923–941.
[34] D. Wu, et al., An end-to-end approach for optimal mode selection in Internet video communication: Theory and application, IEEE J. on Sel. Areas in Comm. 18 (6) (June 2000) 977–995.
[35] Q. Zhang, W. Zhu, Y.Q. Zhang, Resource allocation for multimedia streaming over the Internet, IEEE Trans. on Multimedia 3 (3) (September 2001) 335–339.
[36] R. Zhang, S.L. Regunathan, K. Rose, Video coding with optimal inter/intra-mode switching for packet loss resilience, IEEE J. on Sel. Areas in Comm. 18 (6) (June 2000) 966–976.