9-22-2007

Real-Time Monitoring of Video Quality in IP Networks

Shu Tao
IBM T.J. Watson Research Center

John Apostolopoulos
Hewlett-Packard Laboratories

Roch A. Guérin
University of Pennsylvania, roch@guerinfamily.org

Real-Time Monitoring of Video Quality
in IP Networks

Shu Tao, *Member, IEEE,* John Apostolopoulos, *Senior Member, IEEE,* Roch Guérin, *Fellow, IEEE*

Abstract— This paper investigates the problem of assessing the quality of video transmitted over IP networks. Our goal is to develop a methodology that is both reasonably accurate and simple enough to support the large-scale deployments that the increasing use of video over IP are likely to demand. For that purpose, we focus on developing an approach that is capable of mapping network statistics, e.g., packet losses, available from simple measurements, to the quality of video sequences reconstructed by receivers. A first step in that direction is a loss-distortion model that accounts for the impact of network losses on video quality, as a function of application-specific parameters such as video codec, loss recovery technique, coded bit rate, packetization, video characteristics, etc. The model, although accurate, is poorly suited to large-scale, on-line monitoring, because of its dependency on parameters that are difficult to estimate in real-time. As a result, we introduce a "relative quality" metric (rPSNR) that bypasses this problem by measuring video quality against a quality benchmark that the network is expected to provide. The approach offers a lightweight video quality monitoring solution that is suitable for large-scale deployments. We assess its feasibility and accuracy through extensive simulations and experiments.

Index Terms— Video quality, IP networks, relative video quality, PSNR

I. INTRODUCTION

A Recent study [1] shows video as a fast-growing contributor to Internet traffic, and an increasing number of traditional and emerging video providers are adopting IP as their vehicle for video delivery (e.g., the emergence of IPTV service). As this transition continues, one can expect traffic from video applications to increasingly stress the performance of IP networks. This in turn will affect the quality of the video delivered by those applications, as they are relatively sensitive to network performance fluctuations [14], [2], [3], [6], [5], [27]. As a result, in order to ensure a successful transition to IP-based video, it is key that its quality be consistently comparable to that of traditional video services (i.e., cable or satellite). This calls for an understanding of how IP networks affect video quality, as well as mechanisms that allow real-time, large-scale monitoring of video quality in IP networks [21], [22], [4]. Our goal in this paper is to develop solutions to these two closely related problems.

Video quality is jointly affected by various network-dependent and application-specific factors. For instance, packet losses and delay jitter (which can also translate into losses in the playback buffer) are the major network-dependent factors, while video codec, loss recovery technique, coding bit rate, packetization scheme, and content characteristics are the major application-specific factors affecting video quality and its sensitivity to network errors. Real-time video quality monitoring has two main requirements: (i) the availability of a model that accounts for various network and application parameters and accurately maps them into video quality estimates; and (ii) the ability to easily evaluate these parameters, so as to allow real-time video quality estimation of a potentially very large number of video streams.

Meeting both requirements raises several challenges. On one hand, as described above, generating accurate video quality estimates calls for detailed network and application information. On the other hand, obtaining all these parameters in real-time and in a scalable fashion is difficult. As a result, any practical solution must embody a trade-off between accuracy and real-time usability. While models and systems exist for estimating video quality in packet networks [9], [10], [11], [12], [16], [23], [24], [7], they typically require detailed knowledge of video content and features, and often rely on deep inspection of video packets. Such methods are, therefore, better suited for offline or per-stream video quality estimation, and are not really applicable to real-time, large-scale monitoring of video quality.

Our contributions in this paper are, therefore, two-fold.

First, we develop a model that characterizes the relationship between packet loss and video distortion as a function of specific video codec, loss recovery technique, coding bit rate, packetization, and content characteristics. The proposed loss-distortion model is generally applicable to any motion-compensated video compression scheme, e.g., any MPEG-x or H.26x codec. More importantly, our model is designed in such a way that most of its parameters can be either readily obtained from the application, or easily measured from network paths. To validate our model, we explore two different and practically important video codecs, MPEG-2 and H.264/MPEG-4 AVC, and conduct experiments using a broad range of video content to demonstrate its accuracy.

Second, we leverage our model to develop a video quality evaluation method that does not depend on video content characteristics. Specifically, our model still relies on information unique to each video stream, and in particular requires an accurate estimate of the quality distortion that results from the loss of individual frames or slices (independently decodable

portions of a frame). Traditionally, this information has been either obtained through offline simulations [9], or extracted based on detailed parsing of the video stream [12]. Such approaches are clearly impractical when considering real-time quality monitoring of a large number of video streams, as we expect to be the case in networks that distribute a wide variety of video content to their customers. To address this issue, we introduce a new concept—relative PSNR (rPSNR)—that can be evaluated *independently* of video content characteristics, yet still captures the impact of network impairments on video quality. rPSNR is a metric relative to the quality of a video transmitted on a reference network path whose losses yield acceptable video quality. Using this metric, video quality can be estimated using only network statistics, and basic codec configuration parameters that can all be easily obtained offline, i.e., as a one time operation. We use extensive experiments to demonstrate that rPSNR can provide accurate and real-time video quality estimates across a broad range of network conditions and variations in content characteristics.

The remainder of this paper is organized as follows. We review related works in Section II. Section III then presents the loss-distortion model on which our approach is built and discusses how it captures the impact on video quality of codec selection, coding bit rate, packetization, and video characteristics. In Section IV, we explore qualitatively the accuracy and practicality of the model. Section V introduces the rPSNR metric and demonstrates its effectiveness in path quality estimation. Finally, Section VI concludes the paper with a summary of our findings and possible extensions.

II. RELATED WORKS

A variety of approaches have been developed for assessing the quality of video delivered over packet networks. For example, a straightforward solution for video quality assessment is to directly compare the reconstructed video sequence at the receiver with the original video sequence at the sender [30]. This yields the most accurate assessment of video quality. However, such an approach is unsuitable for real-time, large-scale usage, as it requires the availability of both the received and the original videos. In our context, quality estimation can only take place at either the sender or the receiver, hence making a direct quality comparison infeasible.

A number of other approaches rely on loss-distortion models, i.e., models that map packet losses into video quality (in the form of distortion). Most of these works [9], [10], [11], [12], [16], [23], [24], [7] focus on only a subset of the network and application factors we are trying to account for in this paper. For instance, Stuhlmuller *et al.* [16] modeled distortion in the decoded video sequence as a linear function of the average loss rate. Liang *et al.* [9] extended this work to incorporate the effect of different loss patterns. In contrast to these studies, we seek to develop a model that accounts for *all* major network-dependent and application-specific parameters, including packet losses (in terms of both loss probability and loss burstiness), packetization, type of video codec, video content characteristics, and loss recovery mechanisms. Although our study is not exhaustive so as to

include all possible parameter combinations, it provides a framework that incorporates the impact of these different factors, and we demonstrate its flexibility and effectiveness by testing it under several typical application settings.

The goal of a comprehensive framework for estimating video quality over packet networks was shared by several other works. In particular, the framework of Reibman *et al.* [12], [13] is closest to ours in terms of motivation and approach. In fact, our initial model can be viewed as belonging to its *NoParse* class of methods [12]. Unlike methods in the *Full-Parse* and *QuickParse* classes [12], *NoParse* methods do not rely on deep packet inspection or explicit parsing of the video bit stream. Hence, they have much lower complexity, at the cost of generating less accurate video quality estimates. Our approach also differs from the existing *NoParse* method [12] in two aspects. First, the existing method [12] models video quality as a linear function of loss rate, which is less accurate with bursty losses [9], [12]. Our model is designed to account for both loss rate and loss burstiness, and in particular their impact on the effectiveness of loss recovery mechanisms. This significantly improves the accuracy of quality estimates across loss patterns [18], [19], while remaining considerably simpler than the *FullParse* and *QuickParse* methods. Second, as pointed out by Reibman *et al.* [12], the *NoParse* method requires calibrating the distortion caused by single losses. This calibration is unfortunately dependent on the specific content of the video stream, making it difficult to carry out in real-time. Our loss-distortion model exhibits similar limitations, but we overcome this problem (see Section V) by introducing a new quality metric—rPSNR—that can be estimated independent of video content.

III. LOSS-DISTORTION MODELING

In order to estimate video quality, we need to first investigate the relation between packet losses and distortion in the decoded video. In the following analysis, we use the notation of Liang *et al.* [9], and measure video distortion through the Mean Square Error (MSE). Consider a video sequence with frames of size $N_1 \times N_2$ pixels, $f[k]$ denotes the 1-D vector (of size $N_1 \times N_2$) obtained by line-scanning frame k, and $\hat{f}[k]$ denotes the corresponding frame restored by the decoder. The error signal in frame k is then

$$e[k] = \hat{f}[k] - f[k], \tag{1}$$

which represents the signal impairment in frame k caused by packet losses. The MSE in frame k is defined as

$$\sigma^2[k] = (e^T[k] \cdot e[k])/(N_1 \cdot N_2). \tag{2}$$

The total distortion for a video sequence is the MSE averaged over all its frames. The value of $\sigma^2[k]$ for a given loss event is affected by several network and application-dependent factors. For example, the length of a loss burst determines how many pixels are affected in a frame as well as the number of subsequent frames in which this effect propagates. Additionally, the latter also depends on the number of packets per frame. Conversely, error concealment techniques in the decoder together with the prediction strategy applied by the

encoder and the characteristics of the video itself (i.e., the spatial-temporal correlation between different macro-blocks), also play a role in the resulting distortion in the decoded video.

A. Basic model

An important issue in modeling the distortion that a loss event can cause to predictively encoded video, is the extent to which the resulting error propagates across frames. Specifically, since temporal prediction introduces dependencies between adjacent frames, a single packet loss affects not only the frame with data carried in the missing packet, but also other frames with coding dependencies on it. Fortunately, because of the explicit or implicit spatial filtering applied at the decoder (which can be modeled as a low pass filter [16]), the error signal introduced by a lost packet tends to decay over time. If an error results in an MSE of $\sigma^2[k]$ in frame k, the power of the propagated error in frame $(k + i)$ can be approximated as [9]:

$$\sigma^2[k + i] = \sigma^2[k]\gamma^i. \tag{3}$$

The attenuation factor γ ($\gamma < 1$) accounts for the effect of spatial filtering, and is therefore dependent on the power spectrum density of the error signal and the spatial filtering applied by the decoder, i.e., varies as a function of the video characteristics and decoder processing.

To limit error propagation, periodic intra coding is often used in video compression. As a result, errors in one frame only propagate until the corresponding macro-blocks (or the entire frame) are refreshed by intra coding. For instance, if $(T-1)$ frames are predictively coded (P-frames[1]) between two consecutive intra-coded frames (I-frames), the total distortion caused by losses in frame k is

$$D = \sum_{i=0}^{x-1} \sigma^2[k + i], \tag{4}$$

where x is the number of frames from where the original loss occurred (frame k) to the next I-frame.

This simple model can then be used to evaluate the distortion caused by losing a single slice (a block of independently coded pixels) in a frame. Assuming that the expected initial distortion caused by a lost slice in a frame is σ_S^2, and that within a predictively coded group of frames the location x of the frame with the lost slice is uniformly distributed in $[0, T-1]$, the total average distortion caused by losing a single slice is given by

$$
\begin{aligned}
D_1 &= \sum_{i=0}^{T-1} \sigma_S^2 \gamma^i (1 - \frac{i}{T}) \\
&= \frac{\gamma^{T+1} - (T + 1)\gamma + T}{T(1 - \gamma)^2} \sigma_S^2 \\
&= \alpha \sigma_S^2, \tag{5}
\end{aligned}
$$

where α is a function of γ and T, and accounts for the total propagation effect of the error signal.

[1]To simplify the analysis, we do not consider bi-directionally predicted frames (B-frames) in our model.

Because in IP networks, video data losses are in the form of packets instead of slices, the next step in our modeling effort involves mapping lost packets to lost slices. When losing n ($n \geq 1$) consecutive packets in a single loss event, $f(n)$ slices will be affected, where $f(n)$ is a mapping from the number of lost packets to the number of lost slices. This mapping is a function of both the implementation of the codec and the loss recovery technique [25], [26]. For instance, if each packet contains exactly one slice *and* the decoder simply skips decoding of the slices contained in the lost packets, then $f(n) = n$; however, if a decoder discards an entire frame whenever a single one of its packets is lost, the mapping $f(n)$ takes on a very different form, as discussed in Section III-B. Nevertheless, for any given codec $f(n)$ can typically be computed, and once known, the overall distortion caused by n consecutive packet losses can then be modeled as proportional to the distortion caused by an individual slice loss, i.e.,

$$D_n = f(n)D_1. \tag{6}$$

As previously studied [9], [19], this additive model may slightly underestimate the distortion in the case of bursty losses. However, it greatly simplifies the final model. More importantly, as we show later, this simplification enables us to develop a video quality metric that is independent of individual video characteristics.

The final step in our modeling is to capture the average distortion as a function of loss patterns, and in particular the duration and spacing of error bursts. We use P_n to denote the probability of having n consecutive packets lost in a loss event, and P_m to denote the probability that two consecutive loss events are m packets apart (from the starting packet of the first loss event to that of the second loss event). We assume that each frame is transmitted using L packets, and that n and m are independent random variables. Then, the expected MSE of the reconstructed video can be computed as

$$\overline{D} = \frac{\sum_n P_n D_n}{\sum_m P_m (m/L)} = \frac{\overline{f(n)}}{\overline{m}} L D_1, \tag{7}$$

or equivalently,

$$\overline{D} = P_e \overline{f(n)} L D_1, \tag{8}$$

where P_e is the probability of loss events (of any length) in the video stream; $\overline{f(n)}$ is the average number of slices affected by a loss event. In this modeling, P_e and $\overline{f(n)}$ capture the characteristics of the loss process seen by the video stream ($\overline{f(n)}$ is also affected by packetization and the loss recovery techniques used at the decoder), while L and D_1 are specific to the codec and the video content. For instance, L is typically larger when video is coded at a higher bit rate, and D_1 is itself dependent on α and σ_S^2, as indicated in Eq. (5).

B. Modeling the impact of different codecs

Although most video compression standards support picture segmentation in the form of slices, different codecs react differently to slice losses. For illustration and comparison purposes, we study an MPEG-2 codec and an H.264 codec that have different error handling capabilities. In the MPEG-2 codec, packet losses are handled as follows: If the decoder

detects any number of packet losses in a frame, it discards the entire damaged frame and replaces it with the previously-decoded frame. The H.264 codec employs more sophisticated error-concealment techniques: All received slices are decoded, while the slices contained in the lost packets are recovered using the corresponding slices in the previous frame and the motion-compensation information of the other slices in the same frame.

The above two codecs are likely to result in rather different loss-distortion models, because of the different mappings from packet losses to slice losses, as captured by their respective $f(n)$ values. In the MPEG-2 codec, a loss event affects not only the slices contained in the lost packets, but also the other slices in the same frame, while in the H.264 codec, only the slices in the lost packets are affected. As a consequence, the value of $\overline{f(n)}$ of the MPEG-2 codec tends to be larger than that of the H.264 codec, even if they experience the same loss process. Note that the above descriptions of MPEG-2 and H.264 represent specific implementations. Some MPEG-2 based systems incorporate more sophisticated loss conceal-ment schemes similar to those used by our sample H.264 codec. Conversely, some H.264 systems use only simple loss handling schemes, as basic MPEG-2 systems do. However, we believe that the above examples of MPEG-2 with a simple loss recovery scheme and H.264 with a more sophisticated loss recovery scheme, are representative of many systems either deployed or being deployed and of the applications for which they are used.

For the sake of analytical simplicity, we assume that each video packet contains s slices, and that each video frame is transmitted using L packets[2]. In addition, we also assume that in each frame the starting point of a loss event (when it occurs) is uniformly distributed between the first and the last packets. Under these assumptions, it is possible to derive $f(n)$ for both codecs.

For the MPEG-2 codec, let $r = n \bmod L$, $f(n)$ is then given by

$$f(n) = sL \left[\frac{1}{L}\frac{n}{L} + \left(1 - \frac{1}{L}\right)\left(\frac{n}{L} + 1\right) \right]$$

if $r = 0$, and

$$f(n) = sL \left[\frac{L - r + 1}{L}\left\lceil \frac{n}{L} \right\rceil + \frac{r - 1}{L}\left(\left\lceil \frac{n}{L} \right\rceil + 1\right) \right]$$

if $r \geq 1$, which in both cases simplifies to a linear function of n:

$$f(n) = s\left(n + L - 1\right). \tag{9}$$

For the H.264 codec, the mapping is simply

$$f(n) = sn, \tag{10}$$

since each packet loss causes the loss of s slices. Combining Eqs. (8), (9) and (10), the overall distortion caused to a video sequence by n consecutive packet losses can be modeled as

$$\overline{D} = \begin{cases} s\left(\overline{n} + L - 1\right)P_e L D_1 & : \quad \text{MPEG-2} \\ s\overline{n}P_e L D_1 & : \quad \text{H.264} \end{cases} \tag{11}$$

[2]The model can be extended to accommodate more general video trans-mission schemes, e.g., when s or L are random variables [31], [32], [28], [29].
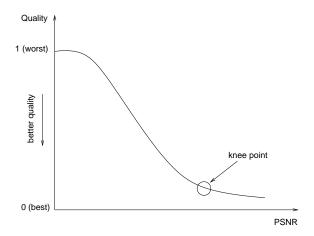

Fig. 1. Mapping between PSNR and video quality as suggested in [30].

Note that the above model captures the effect of (1) the packet loss patterns as expressed by $\overline{n}$ and P_e, (2) the transmission bit rate as expressed by the required number of slices per frame (given by sL), (3) the packetization strategy as expressed by L, (4) the video codec and loss recovery mechanisms as captured through the expression of $f(n)$ for MPEG-2 and H.264, and (5) the video content sensitivity to errors as incorporated in D_1.

Once the distortion ($\overline{D}$) has been captured, the resulting video quality can be characterized using the conventional measure of Peak Signal-to-Noise Ratio (PSNR) [30], i.e.,

$$\text{PSNR} = 10\log_{10}\frac{255^2}{\overline{D}}. \tag{12}$$

PSNR is, however, not a metric that directly measures human perception of video quality. Rather, it can be mapped to a subjective video quality index varying from 0 (best quality) to 1 (worst quality), as suggested by the Video Quality Experts Group (VQEG) [30] and illustrated in Fig. 1. The relation between the two is non-linear and of the form

$$\text{Quality} = \frac{1}{1 + \exp\left(b_1(\text{PSNR} - b_2)\right)}, \tag{13}$$

where $b1$ and $b2$ are parameters that need to be adjusted as a function of video characteristics. Eq. (13) suggests that PSNR only reflects subjective video quality in a certain range. For instance, when its value is less than the one corresponding to the "knee point" of Fig. 1, PSNR has indeed a mostly linear relationship to quality. However, once the PSNR value exceeds that associated with the knee point, subjective video quality essentially "saturates", so that further increases in PSNR do not translate into video quality improvements that are perceiv-able to the human eye. In Section V, we further discuss how to utilize this mapping between PSNR and perceptual video quality to derive practically meaningful quality estimates.

IV. ASSESSMENT OF THE MODEL

Although the above loss-distortion model explicitly incorpo-rates parameters that account for the effect of both network and application factors on video quality, its accuracy still needs to be verified. In this section, we use simulations and experiments to explore the accuracy of the model in characterizing the
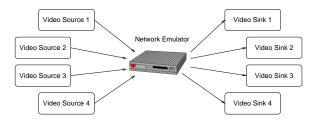
Fig. 2. The network emulator used to simulate path performance variations

Loss rate	Loss pattern	Format	s	L	P_e	$\overline{n}$
2%	Bernoulli	QCIF	2	1	0.020	1.02
			1	2	0.020	1.01
		CIF	2	4	0.020	1.02
			1	8	0.019	1.02
	Bursty	QCIF	2	1	0.018	1.05
			1	2	0.017	1.17
		CIF	2	4	0.014	1.43
			1	8	0.010	2.08
4%	Bernoulli	QCIF	2	1	0.038	1.06
			1	2	0.039	1.05
		CIF	2	4	0.038	1.04
			1	8	0.038	1.04
	Bursty	QCIF	2	1	0.040	1.05
			1	2	0.036	1.13
		CIF	2	4	0.028	1.43
			1	8	0.020	2.06

impact of various application and network parameters on video quality. We also point to limitations of the model as it stands, which we use to motivate our introduction of the relative PSNR (rPSNR) metric in Section V.

A. A qualitative assessment

We carry out a qualitative assessment of the model, with a focus on its ability to capture basic relationships between video quality and loss patterns, especially as a function of packetization schemes and types of video codecs.

One important consideration is to determine if the model's simple mapping, $f(n)$, from packet losses to slice losses is sufficient to account for the interplay between packetization and the codec's error concealment strategy. A related issue is whether Eq. (11), which accounts for different loss patterns only through the loss rate and burstiness as represented by $\overline{n}$ and P_e, is adequate to capture across different codecs the impact of the full loss statistics experienced by the associated videos. In particular, video streams using, say, a different number of packets per frame (L), sample network paths differently, and hence experience different packet-level loss processes. For instance, as shown by Tao *et al.* [20], streams with larger L values tend to see longer packet loss bursts than those with smaller values.

For this initial assessment, we rely on the MPEG-2 and H.264 codecs introduced in Section III-B. For both codecs, we also vary frame size and format to investigate the sensitivity of the model to these factors. In particular, we use two common frame formats: QCIF and CIF. A QCIF frame has 144×176 pixels, while a CIF frame has 288×352 pixels. Correspondingly, each QCIF frame contains 2 slices, while each CIF frame is composed of 8 slices.

Our experimental setting is as shown in Fig. 2. It consists of video sources transmitting packetized video to receivers through a Linux-based network emulator, where packets are dropped based on configured path characteristics (i.e., similar to the *dummynet* tool [15]). The emulator implements paths alternating between two levels of congestion using a simple two-state Markov model [20]. The time that an emulated path stays in the two congestion states, states 0 and 1, is exponentially distributed with respective means of λ_0 and λ_1. In states 0 and 1, the emulator drops packets with probability b_0 and b_1, respectively. Paths with different packet loss rates and burstiness can be emulated simply by varying b_0, b_1, λ_0, and λ_1. We have emulated paths with a wide range of loss rates and burstiness, but only report here on a representative subset. Specifically, we present results for two sets of loss

rates of 2% and 4%, where for each loss rate we include two levels of burstiness: Bernoulli (i.e., $b_0 = b_1 = 0.02$ or 0.04) and bursty (i.e., $b_0 = 0$ and $b_1 = 0.9$). For bursty losses, we select λ_0 and λ_1 so as to generate the desired target loss rate.

For this test, we used a 10-second video sequence, *Foreman*, coded in both QCIF and CIF formats by the two codecs. Our implementations of the MPEG-2 and H.264 codecs are based on their reference code libraries [33] and [34], respectively. For each combination of codec and frame format, we also vary the packetization parameters, i.e., combinations of s and L values, that determine how slices are packaged into network packets. The video streams are transmitted simultaneously through the network emulator, and their respective loss statistics, i.e., P_e and $\overline{n}$, are measured at the receivers, which also decode the received video sequences and compute the corresponding MSE values.

We repeated each experiment for 30 runs. The packet loss statistics and application configuration parameters for all measurements are summarized in Table I. Fig. 3 shows the average, minimum and maximum (shown as error bar) distortions measured from each experiment. Based on these results, we make the following observations.

- *The number of lost packets is the dominant factor affecting video quality in each video configuration.* Specifically, for a give packetization scheme (represented by s and L), frame size, codec selection, and loss burstiness (i.e., $\overline{n}$), the distortion is proportional to the loss event probability P_e, which is consistent with the general form of Eq. (11). For instance, for the MPEG-2 encoded QCIF video with $s = 2$, $L = 1$ and Bernoulli losses, a 2% loss rate corresponds to a distortion of 48.5, while a 4% loss rate corresponds to a distortion of 95.0.
- *The impact of packetization is codec dependent.* In general, the performance of the MPEG-2 codec degrades as L increases[3]. This is because a single lost packet affects not only the slices in that packet, but also all the other

---

[3]Note that the total number of slices in a frame, sL, remains constant.
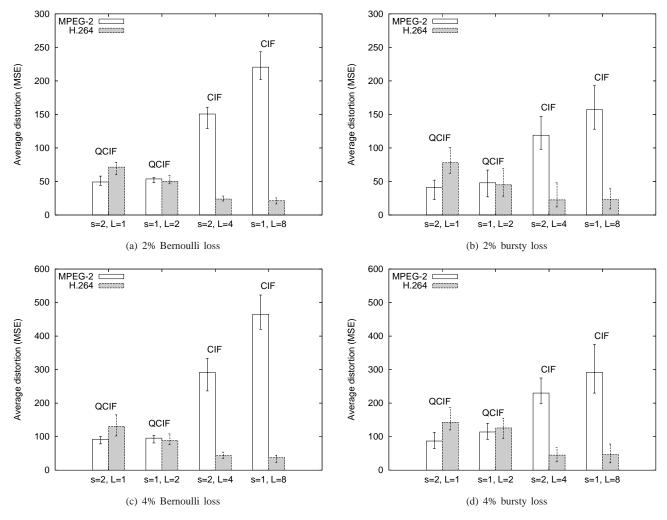
Fig. 3. Video distortion in different loss conditions and with various application configurations.

slices in the same frame. As predicted by Eq. (9), this effect grows with L. The situation is quite different for the H.264 codec, for which Fig. 3 shows that varying L only has a minor effect on video quality. This is consistent with Eq. (11), which states that since for a given frame format and loss rate, the expected number of lost slices per frame should remain constant, so should the resulting distortion.

- *The error concealment scheme determines the codec's sensitivity to loss patterns.* As illustrated in Fig. 3, the more sophisticated slice-level interpolation scheme of our H.264 codec is sensitive to loss burstiness, which typically degrades its error-concealing capability. This is because bursty losses affect a larger number of slices in a frame, which makes it less likely that lost slices can be accurately extrapolated from the received ones. For the MPEG-2 codec, although the above effect still exists, it is over-shadowed by the multiplicative effect that the coarser frame-based error concealment mechanism has on losses. Specifically, we see from Eq. (9) that whether one or all packets in a frame are lost, the resulting number of lost slices is the same. Hence, for a *given* packet loss rate, it is preferable to "group" packet

losses in a single frame rather than spread them across multiple frames. As a result, codecs that like our MPEG-2 codec use a coarse frame-level error concealment scheme perform better (exhibit lower quality distortion) under bursty losses than under Bernoulli losses for a given loss rate. This is again illustrated in Fig. 3[4].

- *The effect of video frame size is function of the error concealment scheme.* Fig. 3 shows that for the same loss rate, the error concealment scheme of the H.264 codec performs better for the CIF format than for the QCIF one. With bigger frames, the lost packets/slices account for a smaller fraction of an entire frame, which facilitates error concealment. On the other hand, as predicted by Eq. (9), the coarse error concealment scheme of the MPEG-2 codec exhibits the opposite behavior. This is because the multiplicative factor $(\overline{n}+L-1)$ is higher for larger frame sizes, as each loss event is converted into a larger number

[4]Note that the figure shows the H.264 codec performing worse than the MPEG-2 codec even in the presence of full frame losses (e.g., when $L = 1$). This is because the motion-compensated concealment technique of the H.264 reference software [34] does not perform well when a complete frame is lost. An improved concealment algorithm would switch between the motion-compensated and previous frame concealment techniques in the presence of full frame losses, and avoid this problem altogether.

of lost slices.

The above observations provide some level of validation of Eq. (11) and of the fact that our model indeed captures, at least qualitatively, the many interactions that relate network losses, codec configuration and types, and video quality.

B. Limitations of the model

Recall that the main purpose for developing a loss-distortion model, as captured in Eq. (11), was to enable accurate, real-time video quality monitoring. Realizing this goal calls for more than just a model. It also requires that all the parameters used in the model be either readily available or easily measurable in real-time.

In order to assess whether Eq. (11) satisfies this requirement, we re-write it as consisting of three terms, as shown in Eq. (14).

$$\overline{D} = (sL) \times \psi \times D_1. \qquad (14)$$

The first term, sL, accounts for the impact of packetization. Both of its parameters, s and L, can be readily obtained from the codec configurations. The second term, denoted as the loss factor ψ in Eq. (14), captures the combined effect of the loss pattern experienced by the video stream, i.e., as measured through $\overline{n}$ and P_e, and the codec's error concealment mechanism:

$$\psi = \begin{cases} (\overline{n} + L - 1) P_e & : \quad \text{MPEG-2} \\ \overline{n} P_e & : \quad \text{H.264} \end{cases} . \qquad (15)$$

For the MPEG-2 codec, ψ is a function of L (which is determined by the packetization scheme), as well as $\overline{n}$ and P_e, which are the only two parameters involved in ψ for the H.264 codec. Since both $\overline{n}$ and P_e can be readily monitored in real-time, ψ can be estimated for both codecs.

Unfortunately, obtaining the last term, $D_1 = \alpha \sigma_S^2$ as per Eq. (5), which represents the level of distortion introduced by a single slice loss, is not easy. Estimating the value of D_1 on-line is challenging because it is not only a function of codec implementation, but also highly dependent on video characteristics. As reported by Stuhlmuller *et al.* [16], the value of σ_S^2 depends on the power spectrum density of the error signal caused by a slice loss, and the strength of loop filtering in the decoder. In general, videos with higher motion make it more difficult to infer the missing data and thereby conceal the losses. Consequently, the distortion caused by a slice loss also tends to be higher for high-motion video.

To illustrate this effect, we plot in Fig. 4 the distortion caused by each individual lost slice in two video sequences, *Foreman* and *Mother & Daughter*, which exhibit high and low levels of motion, respectively. It is clear from the figure that D_1 is typically higher for *Foreman* than for *Mother & Daughter*. It can also be observed that the value of D_1 varies even within a video sequence. For instance, the error signal caused by the loss of an I-slice (e.g., slice number 15, 45, and 75) is typically stronger than that caused by the loss of a P-slice. Furthermore, slices in the same frame may also have different importance in video decoding. For instance, in *Mother & Daughter*, losing the first slice in a frame typically
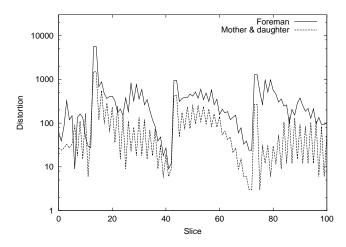

Fig. 4. The value of D_1 for slices in QCIF video *Foreman* and *Mother & Daughter*. Each frame is segmented into 2 slices. Both video sequences are encoded with an intra-coding period of 15 frames. The results for 50 frames (100 slices) are shown in the plot.

causes more distortion than losing the second slice, since there is typically more motion in the top half of its frames.

These observations indicate that video quality estimates depend on the *specific characteristics* of individual videos. Hence, on the same path and with the same loss process, the quality of two received videos may be significantly different. Moreover, for the same video sequence, even if path conditions remain unchanged, quality could vary with scene changes. As a result, estimating *absolute* video quality on a path calls for dynamically assessing the impact of video characteristics (D_1). This is nontrivial [9] and typically requires parsing or decoding the transmitted video bit stream. In practice, carrying out such processing in real-time and for a large number of streams is very challenging, if not impossible. This means that our model, as captured in Eq. (14), still falls short of our initial target of realizing accurate, real-time video quality monitoring. Clearly, achieving this goal requires removing from the model any dependencies associated with individual video characteristics.

V. PRACTICAL VIDEO QUALITY ESTIMATION

In this section, we introduce the concept of a *relative* quality metric, rPSNR, that builds on the model of Eq. (11), but circumvents the problem of dependencies on individual video characteristics. We demonstrate that rPSNR is a metric suitable for real-time, large-scale monitoring of video quality and evaluate its accuracy along multiple dimensions. We also provide additional evaluations based on experiments over a wide-area testbed and with different types of codecs and videos.

A. Relative video quality metric

The basic idea behind rPSNR is to estimate the quality of video transmission on a given path not as an absolute metric, but as one that is relative to some other path with known performance. In Section V-D, we discuss further the characteristics of this *reference* path, as captured in the parameters, $\overline{n} = n^0$ and $P_e = P_e^0$. It typically corresponds to a

path whose performance is known to yield acceptable video quality, e.g., based on benchmarking and provisioning by the network service provider. The relative path quality, or rPSNR, is defined as the difference between the actual PSNR and the target PSNR (i.e., the PSNR of the transmitted video on the reference path). In other words, rPSNR measures how far a path is, quality-wise, from the reference path.

The motivation behind the use of such a relative metric is that by comparing performance along two paths, dependencies on individual video characteristics actually cancel out. Specifically, assume that the loss performance of the reference path is characterized by parameters $\overline{n} = n^0$ and $P_e = P_e^0$, and that the actual path over which the video is being transmitted exhibits loss performance with parameters $\overline{n} = n'$ and $P_e = P_e'$. Let $\overline{D} = D'$ represent the actual video distortion on the current path, and $\overline{D} = D^0$ the video distortion on the reference path. Combining Eq. (12) and Eq. (11), rPSNR is then given by

$$
\begin{aligned}
\text{rPSNR} &= 10 \log_{10} \frac{255^2}{D'} - 10 \log_{10} \frac{255^2}{D^0} \\
&= 10 \log_{10} \frac{\psi_0}{\psi'} \\
&= \begin{cases} 10 \log_{10} \frac{(n^0+L-1)P_e^0}{(n'+L-1)P_e'} & : \text{MPEG-2} \\ 10 \log_{10} \frac{n^0 P_e^0}{n' P_e'} & : \text{H.264} \end{cases}
\end{aligned} \quad (16)
$$

From the above equation, we see that rPSNR no longer depends on the third parameter D_1 of Eq. (14), thus can be computed solely based on estimates for the parameters n^0, P_e^0, L, n', and P_e'. The quantities n^0 and P_e^0 are predefined, the value of L is easy to determine based on application configurations, and estimates for n' and P_e' can be obtained through simple network measurements. For example, monitoring software can be installed in the client devices, e.g., set-top boxes, to collect loss statistics. Alternatively, if this is not feasible, solutions based on polling or lightweight probing, e.g., the method proposed by Tao *et al.* [20], can be used to acquire the necessary information. In the rest of the paper, we assume that accurate estimates are available for all the parameters of Eq. (16).

B. Linear relation between loss factor and distortion

The main reason for rPSNR's independence on video specific factors is the assumption in Eq. (14) that the average distortion, $\overline{D}$, experienced by a video stream is proportional to the loss factor ψ. Therefore, our first step in assessing the validity of the rPSNR model is to verify this linear dependency. This is carried out through a set of simulations and measurements. Specifically, we measure the actual value of $\overline{D}$ for a video stream under various loss conditions that correspond to increasing values of the loss factor ψ. These different loss conditions were generated using the Gilbert-Elliott (GE) model [8] (see Fig. 5), as it allows a simple but systematic exploration of loss processes with different loss rates and loss burstiness. In the GE model, two parameters, p and q, control the loss event probability and loss burstiness. The steady state loss event probability is given by
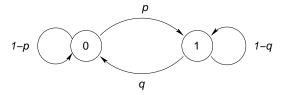
$$
P_e = \frac{pq}{p+q}, \quad (17)
$$

Fig. 5. The GE model used for simulating different loss conditions. State 1: loss; state 0: no loss.

while the average loss burst length is equal to

$$
\overline{n} = 1/q. \quad (18)
$$

In our evaluation, we vary the value of p in $[0, 0.2]$ and the value of q in $[0.6, 1]$ for a total of 400 different combinations or loss processes. Correspondingly, the loss event probability varies from 0 to 0.167, while the loss burstiness varies from Bernoulli (i.e., $\overline{n} \approx 1$) to bursty ($\overline{n} = 1.67$). We apply each simulated loss process to the packet sequences of two video clips, *Foreman* and *Mother & Daughter*, and measure the resulting distortion in the decoded frame sequence. Each video sequence consists of 300 frames and is transmitted in both QCIF and CIF formats. As before, each QCIF frame contains 2 slices, while each CIF frame contains 8 slices. In both cases, each slice is transmitted in a separate packet. The QCIF and CIF sequences have the same frame rate of 30 frames per second, but different bit rates of 100 kb/s and 500 kb/s, respectively. In addition, the two video sequences are coded using both the MPEG-2 and H.264 codecs to investigate the impact of different codecs.

Fig. 6(a) is a scatter plot of the distortion as a function of the loss factor ψ, for both videos in CIF format and using H.264 coding. Fig. 6(b) shows the same results for the videos in QCIF format with MPEG-2 coding. Similar results were obtained for the other combinations of video format and codec. As can be seen from the figures, the average distortion can indeed be modeled reasonably well as a linear function of ψ, as predicted by Eq. (14).

We also observe that the results for the CIF videos generally exhibits a tighter agreement with a linear fit than those for the QCIF videos. This is because, while in our simulations the number of frames is the same in both formats, a CIF video frame requires 4 times as many packets as a QCIF video frame. As a result, the QCIF video stream samples the loss process less frequently than the CIF video stream, and therefore experiences fewer loss events. The smaller number of samples translates into statistically larger variations (or smaller confidence interval) in the resulting distortion measures.

C. Robustness of the metric

We further extend the previous investigation to focus on measuring the accuracy of Eq. (16) in computing actual rPSNR values. Specifically, we select a reference path and study the rPSNR measurement error on 380 different paths that are again generated using the GE model of Fig. 5. In this set of experiments, we use values of $p \in [0, 0.1]$, and $q \in [0.5, 1]$, which span a range of common loss conditions in real networks. The loss process for the reference path is selected
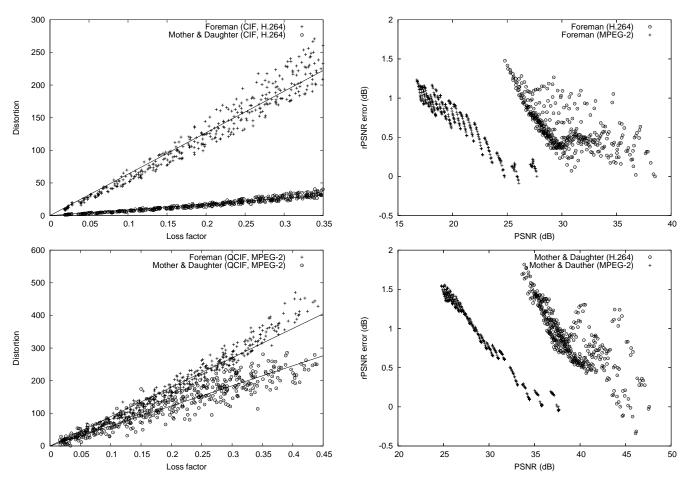
Fig. 6. Video distortion (MSE) as a function of loss factor ψ: (a) the results for videos in CIF format and using H.264 coding (top); (b) the results for videos in QCIF format and using MPEG-2 coding (bottom).

Fig. 7. The accuracy of rPSNR estimation as the function of actual video quality (in PSNR): (a) *Foreman* (top) and (b) *Mother & Daughter* (bottom).

to be Bernoulli with a loss rate of 1% (i.e., $p = 0.01$, $q = 1$)[5]. We use CIF versions of the videos *Foreman* and *Mother & Daughter* in both H.264 and MPEG-2 coding, and simulate the transmission of 3000 frames over the reference path and the 380 paths. We then measure the PSNR of each one of the decoded frame sequences, which allows us to compute the exact rPSNR value of each path combination. As before, each encoded frame is transmitted using 8 packets ($s = 1, L = 8$).

The first step in the experiments is to measure the PSNR of the two videos over the reference path using both the H.264 and MPEG-2 codecs. For *Foreman*, the resulting reference PSNR values are 38.5 dB in H.264 coding and 27.7 dB in MPEG-2 coding, respectively. For *Mother & Daughter*, the reference values are 39.2 dB in H.264 coding and 34.6 dB in MPEG-2 coding. Similar measurements are then performed for all 380 simulated paths, so that the actual difference in PSNR between each path and the reference path can be evaluated. These *actual* rPSNR values are compared to the *computed* rPSNR values derived from Eq. (16) using the loss statistics measured for each path. Fig. 7 reports the differences between the *computed* and *actual* rPSNR values, as a function of *absolute* video quality as measured through the PSNR of the

video transmitted on each path.

As can be seen from the figure, for most of the simulated loss models the rPSNR estimation error is relatively small, and exhibits roughly similar behaviors for both the MPEG-2 and H.264 codecs. Specifically, as the actual PSNR decreases the rPSNR estimation error tends to increase. For example, a negative slope in the relation between PSNR and the rPSNR error can be observed in Fig. 7. This is primarily because a decrease in actual PSNR is typically associated with an increase in the frequency of loss events. As a consequence, consecutive loss events may interfere with each other. This is not accounted for in our model, which assumes that the distortion effects of loss events are independent of each other. This assumption results in underestimating the resulting distortion, hence overestimation of PSNR [9]. We also note that the H.264 and MPEG-2 codecs do not behave identically when it comes to rPSNR error. This is in part because their error concealment mechanisms interact differently with the various loss processes, and in particular exhibit different sensitivity to loss event frequency and loss burstiness. We explore this aspect further next.

Specifically, we re-plot the rPSNR estimation error as a function of p and q in Fig. 8 to better understand how these two characteristics of the loss process affect the accuracy of rPSNR estimation for different codecs and videos. Note that

[5]We discuss the selection of the reference path in more detail in Section V-D.

(a) *Foreman*, CIF, H.264

(b) *Foreman*, CIF, MPEG-2

(c) *Mother & Daughter*, CIF, H.264
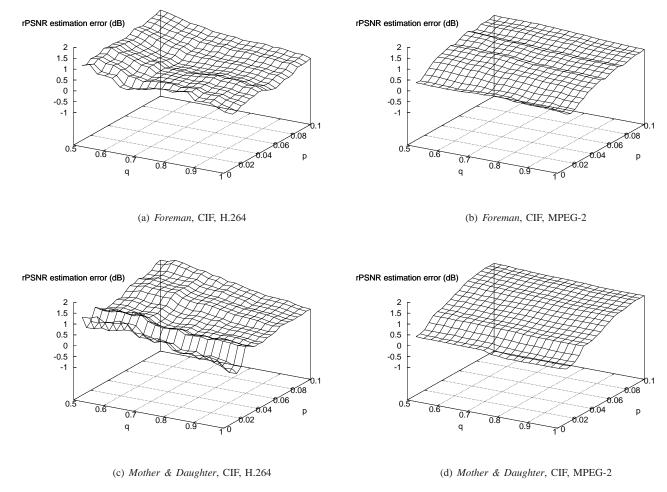

(d) *Mother & Daughter*, CIF, MPEG-2

Fig. 8. The estimation error for rPSNR (relative to the actual rPSNR of the decoded video sequence) as a function of p and q.

for a fixed value of p, increasing q leads to shorter loss bursts, while for a fixed value of q, increasing p leads to a higher loss event probability.

As shown in Figs. 8(a) and (c), for the H.264 codec, the estimated rPSNR tends to become bigger than the actual value as q decreases. This is in part because the error concealment mechanism of H.264 becomes less effective in the presence of multiple close-by losses, which magnifies the aforementioned limitation of the model in properly accounting for consecutive losses. In contrast, as illustrated in Fig. 8(b) and (d), the rPSNR error is relatively insensitive to changes in q when using the MPEG-2 codec. This is again because of how the MPEG-2 error concealment mechanism handles losses. Specifically, the MPEG-2 codec discards entire frames whenever it detects a single loss in a frame. As a result, increasing loss burstiness does not necessarily increase the number of consecutive lost frames. The situation is somewhat reversed when it comes to assessing the impact of p, as while both codecs see higher rPSNR errors as p increases, they exhibit different sensitivities. In particular, the MPEG-2 codec shows a steeper increase in rPSNR error as p increases. This is because a higher frequency of loss events increases the likelihood of consecutive lost frames for MPEG-2.

In summary, while there are differences in how the model performs in estimating rPSNR across different codecs and videos, its overall performance is acceptable and robust to variations in loss characteristics. This is especially true if we limit ourselves to moderate loss scenarios (note that the loss processes we studied have a loss rate up to $p/(p+q) = 16.7\%$). In practice, these are the more likely scenarios, in which accurately assessing how far video quality deviates from its intended target is of significance. Accurately estimating video quality degradation in high loss scenarios is less critical, as classifying those as corresponding to poor (unacceptable) video quality can be done relatively easily, e.g., simply based on loss measurements. In other words, differentiating between bad and very bad video quality is not tremendously meaningful in practice.

D. Selecting the reference path

As discussed earlier, another factor that can affect the accuracy of the rPSNR estimate is the selection of the reference path. Ideally, one should select a reference path whose performance is such that the resulting video is of acceptable quality, i.e., meets certain target requirements. Under such a choice, rPSNR then measures how much worse (or better)

the actual video quality is, compared to its target. However, correctly selecting the reference path conditions is non-trivial. This is in part because, as mentioned earlier (cf. Eq. (13)), video quality is subjective and related to PSNR in a nonlinear fashion. Intuitively, the reference path condition should be selected at the "knee point" of the curve in Fig. 1, so that a positive rPSNR measure indicates that the video quality is good, while a negative rPSNR measure indicates that the video quality is worse than expected in a manner that varies roughly linearly with the estimated rPSNR level.

Nevertheless, finding the knee point on the quality-PSNR mapping curve is not easy. In particular, recall that the values of b_1 and b_2 in Eq. (13) may vary for videos with different characteristics. To solve this problem, we suggest selecting the reference path condition based on the loss factor, which has proved to have a direct impact on video quality. Specifically, an empirical reference value of the loss factor ψ can be defined as a function of the intra-coding period T and the number of packets per frame L. Intuitively, if the loss factor is sufficiently small with respect to the number of packets transmitted in an intra-coding period, the resulting video quality should be reasonably good.

In Fig. 9, we plot the quality of three QCIF video sequences, *Foreman*, *Carphone*, and *Mother & Daughter*, for different Bernoulli loss processes with ψ varying from 1/20 to 1/320. These three video samples cover a reasonably broad range of motion levels: *Foreman* has lots of motion and scene changes; *Carphone* has mild motion and limited scene changes; *Mother & Daughter* is a head-and-shoulder type of video with minimal motion. All video samples were encoded with an intra-coding period (T) of 16 and 32 frames, respectively. In all tests, each frame was transmitted using 2 packets (i.e., $L = 2$). As can be seen from the figure, for all three videos and both codecs, the video samples with $T = 16$ have a quality score close to 0 when $\psi < 1/160$, and conversely when $T = 32$ the quality of the videos saturates (approaches 0) for $\psi < 1/320$. Accordingly, we select a Bernoulli loss process with $\psi = 1/(5TL)$ as the threshold for defining the reference path. We have also tested other loss models using different application parameters, and the results consistently indicate that $\psi = 1/(5TL)$ is a reasonable empirical value to identify the reference knee point.

E. Experimental validations

To further validate the proposed rPSNR methodology for video quality estimation, we conducted extensive experiments on a number of real network paths. Here, we report a set of experimental results collected from a path connecting the University of Pennsylvania (UPenn) and the University of Minnesota (UMN). Our purpose was to validate that our model did not overlook major degradation factors that arose in practice. However, since this path, like most other tested paths, experienced relatively low losses most of the time, we also inserted into the path the network emulator described in Section IV. We let the emulator simulate a channel with two Markov states, with loss probabilities b_0 and b_1, uniformly distributed in $[0, 0.02]$ and $[0.5, 1]$, respectively. The times
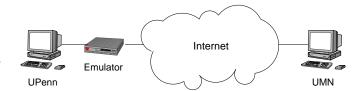

Fig. 10. The setup of loss emulation in our experiments.

spent in each state are exponentially distributed with mean λ_0 and λ_1, which are themselves uniformly distributed between 1 and 10 seconds. The values of b_0, b_1, λ_0, and λ_1 are updated after each state transition. In each experiment, we record the packet traces at the receiver for offline analysis.

As shown in Fig. 10, the sender at UPenn transmits packetized video data through the emulator and the network path to the receiver at UMN. At the receiver, we not only monitor the loss statistics required for evaluating the loss factor ψ (hence estimating rPSNR), but also decode and record the frame sequence that would have actually been seen by the user. Thus, we can compute the rPSNR from Eq. (16) and compare it to the actual rPSNR value derived from the decoded frame sequence[6]. In order to obtain the actual rPSNR, we first measure the PSNR value of the tested video transmitted over the reference path, and then calculate the difference between the PSNR of each decoded frame sequence and this reference value.

We first use a 10-second H.264 video clip (containing 300 frames) from the movie *Dark*, which has a mixture of high motion and low motion scenes. Fig. 11 reports the rPSNR comparison for a 1000-second period, during which the video sequence was repeatedly transmitted. The rPSNR estimation and measurement were performed every 10 seconds. As shown in the figure, the proposed method generates reasonably accurate estimates of quality variations of the transmitted video, except in some instances of fairly good video quality (i.e., periods of rare losses). This can be explained as follows.

Our loss-distortion model relies on averaging the impact of losses across the various (slice and frame) locations where they may occur. Hence, the resulting rPSNR estimate is most accurate when two conditions are met: (i) the average loss burst length and loss event probability can themselves be accurately estimated; (ii) the characteristics of the video content remain relatively constant over the measurement interval, so that averaging over that time frame is meaningful. When the estimation interval is relatively short, say, 10 seconds, content characteristics are likely to remain constant but deriving accurate loss statistics will be difficult. This is particularly so when loss events are rare. As Fig. 11 shows, there is typically a greater difference between the estimated rPSNR and the actual value when the latter is close to 0, i.e., loss events are so rare that video quality is near perfect. In this example, the estimated rPSNR on average deviates 2.5 dB from its actual value.

To satisfy condition (i), we can use longer estimation intervals. However, this could conflict with condition (ii), as well as decrease the responsiveness of the quality monitoring

---

[6]Based on our analysis in Section V-D, we use a Bernoulli loss process with loss factor $1/(5TL)$ as the reference path condition.

(a) *Foreman*

(b) *Carphone*
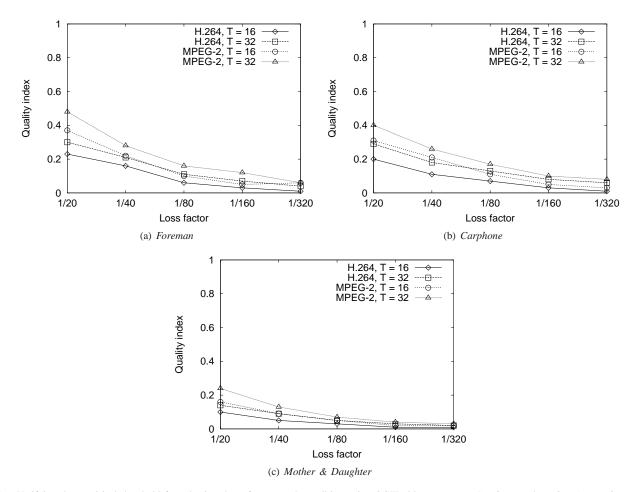
(c) *Mother & Daughter*

Fig. 9.   Verifying the empirical threshold for selecting the reference path condition using QCIF video *Foreman*, *Carphone* and *Mother & Daughter*.
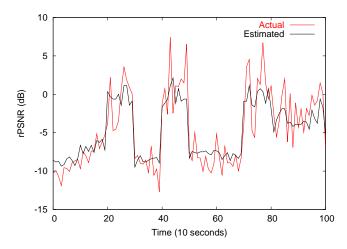

Fig. 11.   Estimating the rPSNR for 10-second video clip *Dark*. Reference path condition: Bernoulli loss process with $\psi = 1/(5TL)$, H.264 coding, 10-second estimation interval.

mechanism to network performance fluctuations. The selection of an appropriate estimation interval is, therefore, a trade-off between these considerations. Our measurement results show that a selection somewhere between a few tens of seconds and a couple of minutes typically yields reasonably accurate video quality estimates. For instance, Fig. 12 shows the results for

the same experiment as that of Fig. 11, but with the estimation interval changed to 1 minute. In this experiment, we use a 1-minute CIF video clip (1800 frames)[7] taken from the same movie *Dark*, and transmit it repeatedly on the monitored path. The estimated and actual rPSNR values for both the H.264 and MPEG-2 video clips are shown in the figure. The figure shows a clear improvement in accuracy. The rPSNR error is now only 1.8 dB and 1.4 dB, for H.264 and MPEG-2 videos respectively. Again, the largest deviations occur in cases where rPSNR is close to or above 0 (i.e., when losses are rare and quality is good). Deviations in these cases are of lesser importance in practice. Meanwhile, in instances when quality is relatively bad, the estimated rPSNR is reasonably accurate; hence providing a good estimate of actual video quality. For instance, focusing on cases where the actual PSNR value is at least 5 dB worse than on the reference path, the rPSNR estimation errors are only 0.9 dB and 0.8 dB for the two codecs.

Besides the video *Dark* used in the above experiment, we have conducted similar tests for other video clips, such as *Foreman*, *Mother & Daughter*, *Highway*, etc. The results

[7]Note that here we use a 1-minute video instead of repeating the 10-second video used in the previous experiment. This is because repeating the same clip multiple times would make the video characteristics artificially more constant, which leads to an unfair comparison in favor of the longer estimation interval.
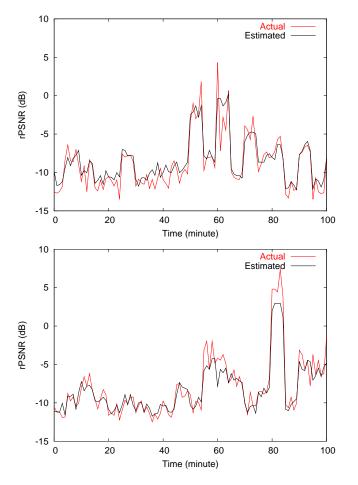
Fig. 12. Estimating the rPSNR for 1-minute video clip *Dark* in H.264 coding (top) and MPEG-2 coding (bottom). Reference path condition: Bernoulli loss process with $\psi = 1/(5TL)$, 1-minute estimation interval.

consistently show that the proposed method is robust and can generate accurate video quality estimates.

VI. CONCLUSION

This paper introduced an approach for real-time video quality monitoring in IP networks. Our goal was to devise a lightweight solution that would allow real-time, large-scale monitoring of video quality. In particular, we wanted to avoid solutions that require detailed knowledge of video characteristics or deep video packet inspection. In that context, our first contribution is the development of a loss-distortion model that accounts for the impact of various network-dependent and application-specific factors on the quality of the decoded video. Our second contribution is in using this model to define a relative video quality metric, rPSNR, that can be evaluated without parsing or decoding the transmitted video bit streams, and without knowledge of video characteristics— thereby significantly reducing complexity while still providing reasonably accurate video quality estimates. The robustness and accuracy of the rPSNR-based method were demonstrated through a broad range of simulations and experiments.

There are several interesting directions for extensions of the method described in this paper. For instance, incorporating the impact of bidirectional coding (B-frames) into our loss-distortion model is clearly of value given the prevalence of its usage in practice. Additionally, exploring video applications with more sophisticated loss concealment (e.g., interpolation) or loss recovery schemes (e.g., Forward Error Correction (FEC)), as well as those using adaptive coding and streaming algorithms, are also interesting directions for future research.

REFERENCES

[1] W. B. Norton. Internet video: The next wave of massive disruption to US peering ecosystem In *Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT)*, Bali, Indonesia, February 2007.

[2] D. Loguinov and H. Radha. Measurement study of low-bitrate Internet video streaming. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.

[3] D. Loguinov and H. Radha. End-to-end Internet video traffic dynamics: Statistical study and analysis. In*Proceedings of IEEE INFOCOM*, New York, NY, June 2002.

[4] Y. Wang, M. Claypool, and Z. Zuo. An empirical study of RealVideo performance across the Internet. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, USA, November 2001.

[5] M Li, M. Claypool, and R. Kinicki. MediaPlayer versus RealPlayer – A comparison of network turbulence. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, November 2002.

[6] F. Li, J. Chung, M. Li, H. Wu, M. Claypool, and R. Kinicki. Application, network and link layer measurements of streaming video over a wireless campus network. In *Proceedings of the 6th Passive and Active Network Measurement Workshop (PAM)*, Boston, MA, April 2005.

[7] P. Frossard and O. Verscheure. Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery. In *IEEE Transactions on Image Processing*, 10(12):1815-1825, December 2001.

[8] W. Jiang and H. Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proceedings of NOSSDAV*, Chapel Hill, NC, June 2000.

[9] Y. J. Liang, J. G. Apostolopoulos, and B. Girod. Analysis of packet loss for compressed video: Does burst-length matter? In *IEEE ICASSP*, Hong Kong, China, April 2003.

[10] A. R. Reibman and V. Vaishampayan. Quality monitoring for compressed video subject to packet loss. In *Proceedings of IEEE ICME*, Baltimore, MD, July 2003.

[11] A. R. Reibman and V. Vaishampayan. Low complexity quality monitoring of MPEG-2 video in a network. In *Proceedings of IEEE ICIP*, Middletown, NJ, September 2003.

[12] A. R. Reibman, V. Vaishampayan, and Y. Sermadevi. Quality monitoring of video over a packet network. *IEEE Transactions on Multimedia*, 6(2):327-334, April 2004.

[13] S. Kanumuri, P. C. Cosman, A. R. Reibman, and V. A. Vaishampayan. Modeling Packet-Loss Visibility in MPEG-2 Video. *IEEE Transactions on Multimedia*, 8(2):341-355, April 2006.

[14] J. M. Boyce and R. D. Gaglianello. Packet loss effects on MPEG video sent over the public Internet. In *Proceedings of ACM Multimedia*, Bristol, England, September 1998.

[15] L. Rizzo. Dummynet. *http://info.iet.unipi.it/luigi*.

[16] K. Stuhlmuller, N. Farber, M. Link, and B. Girod. Analysis of video transmission over lossy channels. *IEEE Journal on Selected Areas in Communications*, 1(6):1012-1032, June 2000.

[17] S. Tao, J. G. Apostolopoulos, and R. Guerin. Real-Time monitoring of video quality in IP networks. In *Proceedings of NOSSDAV*, Stevenson, WA, June 2005.

[18] S. Tao. Improving the quality of real-time applications through path switching. *Ph.D. dissertation*, Department of Electrical and Systems Engineering, University of Pennsylvania, November 2005.

[19] S. Tao and R. Guerin. Application-specific path switching: A case study for streaming video. In *Proceedings of ACM Multimedia*, New York, NY, October 2004.

[20] S. Tao and R. Guerin. On-line estimation of Internet path performance: An application perspective. In *Proceedings of IEEE INFOCOM*, Hong Kong, China, March 2004.

[21] A. C. Dalal and E. Perry. A new architecture for measuring and assessing streaming media quality. In *Proceedings of the 3rd Workshop on Passive and Active Measurements (PAM 2003)*, San Diego, CA, April 2003.

[22] J. van der Merwe, R. Caceres, Y. Chu, and C. Sreenan. Mmdump – A tool for monitoring Internet multimedia traffic. *ACM Computer Communication Review*, 30(4), October 2000.

[23] O. Verscheure, P. Frossard, and M. Hamdi, User-oriented QoS analysis in MPEG-2 video delivery. In *Real-Time Imaging*, vol. 5, 305–314, 1999.

[24] M. Dai, D. Loguinov, and H. Radha. Rate-distortion modeling of scalable video coders. In *Proceedings of IEEE ICIP*, Singapore, October 2004.

[25] C. Perkins, O. Hodson, and V. Hardman. Survey of packet loss recovery techniques for streaming Audio. In *IEEE Network*, vol. 12, 40-48, September 1998.

[26] N. Feamster and H. Balakrishnan. Packet loss recovery for streaming video. In *Proceedings of 12th International Packet Video Workshop*, Pittsburgh, PA, April 2002.

[27] A. Basso, G. Cash, and M. Civanlar. Transmission of MPEG-2 streams over non-guaranteed quality of service networks. In Proceedings of Picture Coding Symposium, Berlin, Germany, September 1997.

[28] I. Rhee. Error control techniques for interactive low-bit rate video transmission over the Internet. In *Proceedings of ACM SIGCOMM*, Vancouver, Canada, September 1998.

[29] B. W. Wah, X. Su, and D. Lin. A survey of error-concealment schemes for real-time audio and video transmissions over the Internet. In *Proceedings of International Symposium on Multimedia Software Engineering*, Taipei, Taiwan, December 2000.

[30] VQEG. Final report on the validation of objective models of video quality assessment. *http://www.vqeg.org/*, August 2003.

[31] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RTP payload format for MPEG1/MPEG2 video. *RFC 2250*, January 1998.

[32] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. *RFC 1889*, January 1996.

[33] libmpeg2. *http://libmpeg2.sourceforge.net/*.

[34] H.264/AVC Software Coordination. Reference software (version JM 8.6). *http://iphome.hhi.de/suehring/tml/index.htm/*.

John Apostolopoulos (S'91 / M'97 / SM'06) received his B.S., M.S., and Ph.D. degrees in EECS from MIT. He joined Hewlett-Packard Laboratories in 1997 where he is currently a Distinguished Technologist and R&D project manager for the Streaming Media Systems Group. He also teaches and conducts joint research at Stanford University where he is a Consulting Associate Professor of Electrical Engineering. In graduate school he worked on the U.S. Digital TV standard, and received an Emmy Award Certificate for his contributions. He received a best student paper award for part of his Ph.D. thesis, the Young Investigator Award (best paper award) at VCIP 2001, was named "one of the world's top 100 young (under 35) innovators in science and technology" (TR100) by Technology Review in 2003, and was co-author for the best paper award at ICME 2006 on authentication for streaming media. His work on media transcoding in the middle of a network while preserving end-to-end security (secure transcoding) has recently been adopted by the JPEG-2000 Security (JPSEC) standard. He currently serves as vice-chair of the IEEE IMDSP and member of MMSP technical committees, and recently was co-guest editor of a special issue of IEEE Network on "Multimedia over Broadband Wireless Networks", an upcoming special issue of IEEE Journal on Selected Topics in Signal Processing on "Network-Aware Multimedia Processing and Communications", general co-chair of VCIP'06, and technical co-chair for ICIP'07. His research interests include improving the reliability, fidelity, scalability, and security of media communication over wired and wireless packet networks.

Roch Guérin (F'01 / ACM'98) received an engineer degree from ENST, Paris, France, in 1983, and M.S. and Ph.D. degrees in EE from Caltech in 1984 and 1986, respectively.

He joined the Electrical and Systems Engineering department of the University of Pennsylvania in 1998, as the Alfred Fitler Moore Professor of Telecommunications Networks. Before joining Penn, he spent over twelve years at the IBM T. J. Watson Research Center in a variety of technical and management positions. From 2001 to 2004 he was on partial leave from Penn, starting Ipsum Networks (now part of Iptivia), a company that pioneered the concept of protocol participation in managing IP networks. His research has been in the general area of networking, with a recent focus on developing networking solutions that are both lightweight and robust across a broad range of operating conditions.

Dr. Guérin has been an editor for a variety of ACM and IEEE publications, and has been involved in the organization of many ACM and IEEE sponsored activities. In particular, he served as General Chair of the IEEE INFOCOM'98 conference, as Technical Program co-chair of the ACM SIGCOMM'2001 conference, and as General Chair of the ACM SIGCOMM'2005 conference. In 1994 he received an IBM Outstanding Innovation Award for his work on traffic management and the concept of equivalent bandwidth. He was on the Technical Advisory Board of France Telecom for two consecutive terms from 2001 to 2006, and on that of Samsung Electronics in 2003-2004.

Shu Tao (S'03 / M'05) received B.S. and M.S. degrees in Electrical Engineering from Wuhan University, China, in 1997 and 2000, respectively. He received Ph.D. degree in Electrical Engineering from the University of Pennsylvania in 2005.

Dr. Tao joined the IBM T. J. Watson Research Center as a Research Staff Member in 2005. Recently, he has been working on network resiliency enhancement and proactive problem diagnosis in systems and networks. His research interests are in the general area of network systems and applications, including network measurement, performance modeling, peer-to-peer systems, and network support of multimedia applications.