Rate-Distortion Analysis and Quality Control in Scalable Internet Streaming

Min Dai, Dmitri Loguinov, Member, IEEE, and Hayder Radha, Senior Member, IEEE

Abstract—Rate-distortion (R-D) modeling of scalable video encoders has recently become an important issue in video streaming. In the first half of the paper, we model rate-distortion of DCT-based fine-granular scalable coders and derive a simple operational R-D model for Internet streaming applications. Experimental results demonstrate that our R-D formula, an extension of the classical R-D result, is very accurate within the domain of scalable coding methods exemplified by MPEG-4 FGS and H.26L progressive FGS. In the second half of the paper, we examine congestion control and dynamic rate-scaling algorithms that achieve smooth visual quality during streaming. Since the Internet is a changing environment shared by many sources, even R-D based quality control often cannot guarantee non-fluctuating PSNR in variable bitrate (VBR) channels without the help from an appropriate congestion controller. Thus, we apply recent utility-based congestion control methods to our problem and show how they can benefit future streaming applications. To our knowledge, this approach and the subsequent analysis are novel.

I. INTRODUCTION

RATE-DISTORTION (R-D) curves are useful not only in source coding, but also in Internet video streaming. While it is well-known that R-D based compression approaches can adaptively select quantization steps and maximize video quality under given buffer constraints [6], [22], R-D curves can also be used during streaming rate-control to optimally allocate bits in joint source-channel coding [3], [13], avoid network congestion [4], [25], and achieve constant quality [35], [42], [43].

Accurate modeling of R-D curves of real encoders and channel characteristics of real communication systems (e.g., the Internet) is always challenging due to the diversity of source images and the inherent complexity of Internet-like channels [30]. Typically, R-D modeling is undertaken using either the empirical or the analytical approach, each of which has its own benefits and drawbacks. The empirical approach obtains R-D curves by interpolating between \((R, D)\) samples of a given encoder [28]. The analytical approach derives R-D models from the angle of information and/or quantization theory assuming certain (usually simplified) statistical and correlational properties of the source [7], [11], [14]. While the empirical approach usually results in better estimation of the curve, it fundamentally lacks theoretical insight into the structure of the coding system.

Thus, in order to accurately take into account complex statistical structure and source correlation of real encoders, a third, operational, type of R-D models is widely used in practice [4], [11], [12]. An operational R-D model obtains the basic structure of the curve in a closed-form analytical expression, but then parameterizes the equation according to several parameters sampled from the actual system (e.g., [4], [11], [12]).

Although there are numerous applications of R-D modeling in scalable Internet streaming [35], [37], [42], [43], the majority of current R-D models are built for images and/or non-scalable video coders [5], [14]. To overcome this gap in current knowledge of scalable coding R-D systems and provide future video streaming applications with accurate R-D models, this paper derives two operational R-D models based on statistical properties of scalable sources and existing models of bitrate [12]. Our first result applies to the enhancement layer of a variety of scalable DCT coders (including FGS and PFGS) and demonstrates that distortion \(D\) is a function of both \(R\) and its logarithm \(\log R\):

\[
D = \sigma^2_x - (a \log^2 R + b \log R + c)R, \tag{1}
\]

where \(\sigma^2_x\) is the variance of the source and \(a - c\) are constants.

Since this formula is too complicated for time-constrained streaming applications, our second R-D result is a polynomial approximation of (1), which can be summarized as an operational extension of the traditional model \(D \sim 2^{-2R}\):

\[
D = \sigma^2_x 2^{-\alpha R + b \sqrt{R}}, \tag{2}
\]

where \(a < 0\) and \(b\) are constants (different from those in (1)) dependent on the properties of the source. During our journey to obtain these results, we also offer a new model for the distribution of DCT residue and derive an accurate Markov model for bitplane coding (more on this in the following sections).

The second issue that must be addressed in video streaming is quality control. This is an important concern for end users since human eyes are sensitive to quality fluctuation, which is often present in constant bitrate (CBR) coded base layers and in video streaming over variable bitrate (VBR) channels. Thus, during streaming, the server must rely on an efficient R-D model to rescale the enhancement layer to both match the available bandwidth in the network and smooth out visual quality fluctuations introduced by the base layer [35], [42], [43].

While video streaming has strict Quality of Service (QoS) requirements on bandwidth, delay, and packet loss, the current best-effort Internet does not provide any QoS guarantees to

Min Dai and Dmitri Loguinov are with Texas A&M University, College Station, TX 77843 USA (email: min@ee.tamu.edu, dmitri@cs.tamu.edu)
Hayder Radha is with Michigan State University, East Lansing, MI 48824 USA (email: radha@egr.msu.edu)
end flows. Therefore, congestion control is typically the only viable solution that allows streaming applications to avoid substantial packet loss, share the bottleneck routers fairly, and offer a pleasant video quality to end users. Many current congestion controllers for streaming applications are built on top of TCP-friendly schemes and usually exhibit difficulty in maintaining a smooth channel due to their large rate fluctuations [1], [9] and asymptotic instability [40].

We take a different approach and extend the continuous-feedback congestion control methods proposed by Kelly et al. [18]. We study their performance in constant-quality network streaming and show that the resulting controller is stable under arbitrarily delayed feedback and offers end flows exponential convergence to link utilization (contrast this with AIMD’s linear rate of convergence). This is one of the first papers to apply provably stable AQM congestion control [41] in constant-quality video streaming.

The rest of the paper is organized as follows. In Section II, we give a brief overview of related work. Section III provides the big picture of R-D modeling of scalable coders. Section IV derives a simple distortion model for Laplacian sources. Section V models the bitrate of scalable coders and section VI discusses our main operational R-D model. In section VII, we analyze Kelly’s controller and describe our quality control algorithm. Finally, Section VIII concludes this paper.

II. RELATED WORK

In this section, we briefly overview the work related to the statistical nature of video sources, R-D functions, and quality-control during streaming.

A. Source Statistics

Assume that the input to the encoder (e.g., DCT coefficients, DCT residue, or DWT coefficients) are modeled by a random variable $X$. It has been a long-standing problem to determine statistical properties of $X$ that can be applied to a wide variety of images and/or coding methods. Some well-known models for $X$ include the Gaussian [28], the Laplacian [33], and the Generalized Gaussian distribution (GGD) [28]. While many experimental results indicate that the Gaussian distribution is often not very accurate in modeling DCT data, it is generally agreed that the Laplacian distribution provides acceptable accuracy in a variety of DCT-based coding methods [20], [33].

In applications where even higher accuracy is required, the GGD (despite its complexity) is sometimes used to model source data [28]. Recall that the GGD is given by its density function:

$$f(x) = \frac{\alpha \nu}{\Gamma(1/\nu)} e^{-|\alpha x|^{\nu}},$$

(3)

where $\Gamma(.)$ denotes the gamma function, $\nu$ is the shape parameter;

$$\alpha = \frac{1}{\sigma_x} \sqrt{\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}},$$

(4)

and $\sigma_x$ is the standard deviation of the source. For $\nu = 1$, the GGD becomes a Laplacian distribution and for $\nu = 2$, it becomes a Gaussian distribution.

Among other modeling work, Yovanof et al. [39] point out that a single model is usually insufficient to describe statistical properties of complex sources. Eade et al. [8] show that a linear mixture of several distributions offers more degrees of freedom and fits actual samples better. Smoot et al. [33] also mention that the mixture model achieves higher accuracy than a single distribution in modeling DCT coefficients.

B. R-D Modeling

We next describe several closed-form R-D functions commonly used in video coding. Recall that the most well-known R-D result stems from classical Shannon’s work [5] and early developments in quantization theory [30]:

$$D = \sigma_x^2 2^{-2R},$$

(5)

where $D$ denotes MSE distortion, $R$ is the bitrate of the coded sequence, and $\sigma_x^2$ is the variance of the source. While directly applicable only to a small set of sources, this model is still widely used in video streaming [12], [35].

Shannon’s classical model is the basis for many operational R-D models and is often extended to account for non-Gaussian distributions and non-trivial source correlation [11], [12]:

$$D = \gamma \varepsilon^2 \sigma_x^2 2^{-2R},$$

(6)

where $\gamma$ is the correlation coefficient of the data and $\varepsilon^2$ is a source-dependent scaling parameter (1.4 for Gaussian, 1.2 for Laplacian, and 1 for uniform sources).

Distortion depends only on the statistical properties of the signal (i.e., its distribution); however, the rate also depends on the correlation among the input symbols [11], which explains the independent derivations of $D(\Delta)$ and $R(\Delta)$ often used in the literature. For uniform quantizers (UQ), the classical model is often decomposed into two separate models with respect to quantizer step $\Delta$: distortion $D(\Delta)$ and rate $R(\Delta)$. Under uniform quantization, both models can be summarized as [11]:

$$D(\Delta) = \frac{\Delta^2}{\beta}, \quad R(\Delta) = \frac{1}{2} \log_2 \left( \frac{\varepsilon^2 \sigma_x^2}{\Delta^2} \right),$$

(7)

where $\beta$ is 12 for small $\Delta$. To account for a wider range of $\Delta$, parameter $\beta$ typically needs to be empirically adjusted based on samples of the R-D curve or other source parameters [11].

For Laplacian sources with density $p(x) = \frac{1}{2} e^{-|x|}$, the R-D function can be also written in terms of the Mean Absolute Difference (MAD) distortion $D_M$ [34]:

$$R = -\log(\alpha D_M),$$

(8)

where $\alpha$ is some constant. Using Taylor expansion of (8), Chiang et al. [4] propose an operational R-D model for Laplacian sources and apply it to the MSE distortion $D$:

$$R = a D^{-1} + b D^{-2},$$

(9)

where parameters $a$ and $b$ are obtained from samples of the empirical R-D curve.

In another recent development, He et al. [12] propose a unified $\rho$-domain R-D model, in which the bitrate is estimated by a linear function of the percentage of zero coefficients in
each video frame. In this framework, distortion $D$ for each $\Delta$ is computed directly using the DCT coefficients without any modeling.

Besides the above operational models, there are purely empirical ways to estimate R-D curves. Among the numerous studies, e.g., Lin et al. [23] use cubic interpolation of the empirical curve and Zhao et al. [42] apply similar methods to FGS-related streaming algorithms.

C. Quality Control in Streaming

The MPEG-4 standard [22], [31] has recently adopted Fine Granular Scalability (FGS) into its streaming profile and motivated the development of new scalable compression paradigms such as progressive FGS (PFGS) [38]. FGS consists of a single base layer and one enhancement layer, the latter of which contains the residual signal coded using embedded DCT. Due to non-stationary characteristics of video sources (such as scene changes), the base layer often exhibits significant quality fluctuation that needs to be smoothed out by the server, which must properly select (for each frame) the fraction of the FGS layer that should be transmitted over the network.

Many approaches have been proposed to achieve constant quality in video streaming, e.g., Wang et al. [35] use (6) to estimate the R-D curve of PFGS and propose an optimal bit allocation scheme that reduces quality fluctuation based on the estimated R-D curve, the authors of [42] apply a similar method to FGS video, while Zhao et al. [43] obtain the R-D curve empirically and adopt Newton’s search method to achieve constant quality during transmission of video over the Internet.

III. SCALABLE R-D MODELING: THE BIG PICTURE

A. Preliminaries

In scalable streaming applications, R-D curves can be employed to decide the proper scaling of the enhancement layer to both match the channel capacity and achieve smooth video quality at the receiver. Although there are many R-D based streaming solutions [35], [37], [42], [43], to our knowledge, the theoretical foundation behind the shape of R-D curves of scalable coders has not been investigated thoroughly.

In what follows, we first describe our main modeling framework and explain why distortion in the FGS layer is sufficient to model end-user visual quality. Then we study the statistical properties of DCT residue and propose a mixture Laplacian model, which is simpler than the GGD and achieves better results in our test sequences. Subsequently, we derive an R-D function of mixture Laplacian sources and reduce its expression to a simple parameterized closed-form equation.

Assume that the rate of the base layer is $R_B$ and its distortion is $D_B$. Furthermore, assume that the server transmits $R_E$ bits from the enhancement layer and achieves some combined distortion $D$. While the traditional approach is to model the distortion $D$ as a function of the total bitrate $R = R_B + R_E$, several simplifications of this framework are possible. First, since the server during streaming is concerned only with the rate of the enhancement layer, its R-D curves should be expressed in terms of $R_E$. Second, as we show below, enhancement-layer distortion created at the server by discarding several least-significant bitplanes during transmission is sufficient for estimating the actual end-user distortion $D$.

To better understand the above discussion, we illustrate the coding process of FGS in Fig. 1. In the figure, signal $U$ in the spatial domain is transformed (with some round-off errors $\omega_1$) into signal $X$ in the DCT domain. Then $X$ is separated into the base layer $B$ and the enhancement layer $E$ by the encoder (i.e., $B + E = X$). The enhancement layer contains the residual signal, which is necessary to reconstruct the original image from the coded base layer $B$. After losing a certain number of bitplanes in the server module, the residual signal $E$ becomes $\tilde{E}$ and is then added to the base layer at the receiver to produce distorted signal $\tilde{X} = B + \tilde{E}$ in the DCT domain. Finally, $\tilde{X}$ is converted into the spatial domain (with additional round-off errors $\omega_2$) to become $\tilde{U}$, which is displayed to the user.

As shown in Fig. 1, there are three levels of distortion: spatial-domain distortion $D = E[(U - \tilde{U})^2]$, DCT-domain distortion $D_{\text{DCT}} = E[(X - \tilde{X})^2]$, and the “internal” distortion $D_{\text{FGS}} = E[(E - \tilde{E})^2]$ produced by the server during streaming. Also note that DCT/IDCT round-off noises $\omega_1$ and $\omega_2$ shown in the figure are commonly assumed to be insignificant (which is true except in very high bitrate cases) and are often neglected in R-D modeling.

Next, we examine the relationship among these three types of distortion in the FGS coding process. It is well known that in an ideal encoder-decoder system, spatial-domain distortion $D$ and DCT-domain distortion $D_{\text{DCT}}$ are equal. While the next lemma is trivial, we present it here for completeness of presentation.

Lemma 1: Assuming that the encoder computes the residue in the DCT domain\(^1\), distortion $D_{\text{DCT}}$ equals $D_{\text{FGS}}$.

Proof: Recall that $\tilde{X} = B + \tilde{E}$. Noticing that

$$D_{\text{DCT}} = E[(X - \tilde{X})^2] = E[(X - (B + \tilde{E}))^2] = E[(E - \tilde{E})^2] = D_{\text{FGS}}, \quad (10)$$

we get the desired result. \hfill \blacksquare

\(^1\)Note that the lemma holds even if the encoder (despite increased complexity) computes the residue in the spatial domain.
function of enhancement rate $R_E$ (neither of which requires any information from the base layer). Thus, throughout the paper, we model $D_{FGS}$ instead of $D$ since the statistical properties of DCT residue are more mathematically tractable than those of the original signal. Note, however, that we verify our modeling efforts against the actual distortion $D$ observed by the end-user, which results in a slightly worse performance of the models due to the existence of round-off errors between $D_{FGS}$ and $D$.

B. Source Statistical Properties

As discussed earlier, the input to the enhancement layer is the DCT residue between the original signal and the reconstructed image of the base layer [27]. Our first goal is to model the distribution of DCT coefficients in the residual image $E$ since their statistical properties affect the resulting R-D model and are not generally available in the current literature. Throughout the paper, we use MPEG-4 FGS and PFGS to code popular CIF sequences Foreman, Coastguard, Carphone, and Mobile. The base layer is always coded at 128 kb/s and 10 fps, which is a common evaluation setup for R-D analysis of scalable video streaming [35], [42], [43]. While our discussion mainly involves MPEG-4 derivatives of FGS, our analytical results are applicable to a wide range of scalable (embedded) coding method and even non-scalable streams of MPEG-4 and H.264.

Fig. 2 shows Gaussian and Laplacian distributions fitted to the PMF (probability mass function) of FGS residue in frame 0 of CIF Foreman. The left side of the figure demonstrates that the signal is in fact zero-mean; however, neither Gaussian, nor Laplacian distributions fit the center peak. Note that it is important to model the peak of the distribution of embedded visual signals since it often contains a large fraction of the coefficients (in FGS, usually 20% or more). The right side of the figure shows that the Gaussian tail decays too quickly and that the Laplacian distribution cannot model both the peak and the tail simultaneously.

We also studied the suitability of the GGD to model the data and (as expected) found that it was more accurate than the other two distributions (i.e., Gaussian and Laplacian). However, due to its complexity, the GGD does not generally present an analytically appealing alternative to simpler methods. In fact, we show below that there exists a model of DCT residue that is more accurate and yet significantly simpler than the GGD.

Examining the tail of the PMF on a log scale in Fig. 2 (right), it is clear that its shape resembles two straight lines (each of which is an exponential function on a log scale). Similar observations hold for other frames and sequences (not shown here). Building upon these observations and on previously suggested methods for non-scalable DCT modeling [8], we next investigate a linear mixture model of two Laplacian distributions.

Consider DCT residue to be a random variable drawn from two different distributions. With probability $p$, the residue is selected from the Laplacian component with low variance and with probability $1−p$, from the component with high variance. Then the density of $X$ can be written as:

$$p(x) = p \frac{\lambda_0}{2} e^{-\lambda_0|x|} + (1-p) \frac{\lambda_1}{2} e^{-\lambda_1|x|},$$

where $p$ is the probability to obtain a sample from the low-variance model, and $\lambda_0$ and $\lambda_1$ are the shape parameters of the corresponding Laplacian distributions. The parameters of (11) can be optimally estimated using a variety of methods, including the Expectation-Maximization (EM) algorithm [2] used in this work. We next examine the accuracy of this model in real sequences.

Fig. 2 demonstrates that (11) models the same frame 0 of the CIF Foreman sequence with more accuracy than the traditional Gaussian/Laplacian models. In fact, the fit of the mixture model was even better than that of the GGD in all test sequences. We show this result for Foreman and Carphone using the $\chi^2$ statistic in Table I for 10 and 20 bins utilized in the computation of $\chi^2$. In both cases, the table shows that the mixture model produces much smaller errors $\chi^2$ than any of the other models.

C. Classical R-D Models

The classical R-D model $D \sim 2^{-2R}$ is typically obtained under the assumptions of an infinite block length and high-resolution (i.e., small $\Delta$) quantization that allows the PMF of the signal in each $\Delta$-bin to be approximated by a constant [14], [30]. Neither of these two assumptions generally holds in practice, especially in cases of sharply decaying PMF of DCT residue (which is not constant even in small bins) and low-birate streaming (which inherently relies on high $\Delta$).

To better understand some of these intricacies, first notice that the traditional R-D framework (5) converted to PSNR

Table: The Average Values of $\chi^2$ in Test Sequences

<table>
<thead>
<tr>
<th></th>
<th>Bins</th>
<th>Gaussian</th>
<th>Laplacian</th>
<th>GGD</th>
<th>Mixture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carphone</td>
<td>10</td>
<td>$8.2 \times 10^{22}$</td>
<td>$6.9 \times 10^4$</td>
<td>5,756</td>
<td>3,072</td>
</tr>
<tr>
<td>Foreman</td>
<td>10</td>
<td>$1.3 \times 10^{15}$</td>
<td>$6.6 \times 10^4$</td>
<td>3,437</td>
<td>1,939</td>
</tr>
<tr>
<td>Carphone</td>
<td>20</td>
<td>$4.6 \times 10^{26}$</td>
<td>$8.5 \times 10^4$</td>
<td>9,160</td>
<td>5,373</td>
</tr>
<tr>
<td>Foreman</td>
<td>20</td>
<td>$2.5 \times 10^{18}$</td>
<td>$7.9 \times 10^4$</td>
<td>5,735</td>
<td>3,916</td>
</tr>
</tbody>
</table>

Note: $\lambda_0$ and $\lambda_1$ are the shape parameters of the Laplacian distributions. The parameters of (11) can be optimally estimated using a variety of methods, including the Expectation-Maximization (EM) algorithm [2] used in this work. We next examine the accuracy of this model in real sequences.
quality becomes a linear function of rate $R$:

$$PSNR = 10 \log_{10} \frac{255^2}{D} = \frac{20R}{\log_2 10} + 20 \log_{10} \frac{255}{\sigma_x},$$ (12)

As shown in Fig. 3 for two different frames of CIF Foreman, the actual R-D curve of these frames cannot be modeled by a straight line over the entire range of $R$. In fact, even a heuristically selected quadratic curve in the figure (used here only for illustration purposes) is incapable of modeling the entire range of the bitrate. Both models exhibit significant discrepancy reaching as high as 5 dB.

In our next example, we evaluate the accuracy of models (6) and (9), which are extensions and/or improvements of the basic linear model. Fig. 4 shows the R-D curves produced by (6) (labeled as “classical” in the figure) and (9) (labeled as “Chiang et al.”). Note the log-scale of the $x$-axis, which is used here to demonstrate that (9) exhibits better performance and produces negative values of $R$ for sufficiently large $D$ (this cannot be shown in the PSNR figure, so the curve simply stops).

In the following two sections, we analyze the distortion and bitrate of scalable coders separately and then combine our results into a simple operation model in section VI. As shown below, this approach provides a better characterization and understanding of the R-D properties of scalable coders.

### IV. DISTORTION MODEL

#### A. Distortion Model for Bitplane Coding

Recall that in an image/video coder, the distortion mostly comes from quantization errors, even in a lossy predictive coder [11], [36]. A uniform quantizer is widely applied in video coders for its approximate optimality in the high bitrate case [6], which makes model (7) popular due to its simplicity and accuracy under these assumptions. However, when the output rate is not high enough, (7) requires empirical adjustments to $\beta$ to achieve satisfying results [11], [32]. In what follows, we first derive a distortion model for discrete Laplacian sources and then reduce the result to a simple approximation that we use later for streaming purposes.

Now that we conceptually know that the source data are drawn from two Laplacian distributions, we can focus on understanding the properties of distortion $D(\Delta)$ caused by bitplane coding. In the base layer, the distortion comes from applying a uniform (usually) mid-point quantizer to each DCT coefficient (different quantizers are often applied to different frequencies) [10], [11]. On the other hand, embedded coders such as FGS use bitplane coding, in which all coefficients are transmitted bit-by-bit from the most-significant bitplane (MSB) to the least-significant bitplane (LSB). This can be viewed as applying a quantizer step $\Delta = 2^{n-2}$, where $n$ is the total number of bitplanes in the frame and $z$ is the current bitplane number.3 For example, assuming that the maximum DCT coefficient is 40, $n$ is 6 and $\Delta$ takes the values equal to 32, 16, 8, 4, 2, 1 for bitplanes 1 through 6, respectively.

Lemma 2: For Laplacian sources with PMF $p(m) = ae^{b|m|}$, $a > 0$ and $b < 0$, the MSE distortion after uniform quantization with step $\Delta$ is:

$$D(\Delta) \approx \frac{2ae}{1 - e^{-a}},$$ (13)

where $\xi$ is given by:

$$\xi = e^{b(\Delta - 1)} \left( \frac{(\Delta - 1)^2}{b^2} - 2 \frac{(\Delta - 1)}{b^3} + 2 \frac{b}{b^3} \right) - \frac{2}{b^3}$$ (14)

Proof: Since the PMF $p(m)$ of the source is always symmetric, the distortion after bitplane coding can be written as:

$$D(\Delta) = 2 \sum_{k=0}^{N/\Delta} \sum_{m=k\Delta}^{(k+1)\Delta-1} (m-k\Delta)^2 p(m).$$ (15)

where $N$ is the maximum value of the quantizer equal to $2^{n-1}$ (recall that $n$ is the total number of bitplanes). Replacing $m$ with $k\Delta + i$ in (15):

$$D(\Delta) = 2 \sum_{k=0}^{N/\Delta} \sum_{i=0}^{\Delta-1} (k\Delta + i - k\Delta)^2 p(k\Delta + i)$$

$$= 2 \sum_{k=0}^{N/\Delta} \sum_{i=0}^{\Delta-1} i^2 ae^{b(k\Delta+i)}$$

$$= 2a \sum_{k=0}^{N/\Delta} e^{b\Delta} \sum_{i=0}^{\Delta-1} i^2 e^{bi}. $$ (16)

While traditional quantizers implement mid-point reconstruction, bitplane coding can be viewed as a floor function applied to the result. Further note that MPEG-4 FGS has an option for “quarter-point” reconstruction, in which the decoder adds $\Delta/4$ to the result. For brevity, we omit $\Delta/4$ in all derivations; however, it can be shown that our final result holds for quarter-point quantizers as well.

Fig. 3. Frame 39 (left) and 73 (right) in FGS-coded CIF Foreman sequence.

Fig. 4. R-D models (6), (9), and the actual R-D curve for frame 0 in CIF Foreman (left). The same for frame 84 in CIF Foreman (right).
The result in (16) is a product of two summation terms, each of which can be computed separately. First notice that \( \sum i^2 e^{bi} \) is easily estimated using integration:

\[
\sum_{i=0}^{\Delta-1} i^2 e^{bi} \approx \int_0^{\Delta-1} x^2 e^{bx} dx.
\]

Solving (17), we have:

\[
\int_0^{\Delta-1} x^2 e^{bx} dx = e^{bx} \left( \frac{x^2}{b} - \frac{2x}{b^2} + \frac{2}{b^3} \right) \bigg|_0^{\Delta-1} = \xi,
\]

where \( \xi \) is given by (14). Next consider term \( \sum e^{bk\Delta} \) in (16) and notice that it is a geometric series with the following expansion:

\[
\sum_{k=0}^{N/\Delta} e^{bk\Delta} = \frac{1 - e^{b(N+\Delta)}}{1 - e^{b\Delta}} \approx \frac{1}{1 - e^{b\Delta}},
\]

where the last approximation holds since \( e^{b(N+\Delta)} \) is negligible and can be omitted for all practical values of \( N \) and \( b \). Multiplying (19) by \( 2\alpha \) and \( \xi \), we obtain (13).

Notice that when \( \Delta = 1 \), (13) produces \( D = 0 \) and when \( \Delta = \infty \), the distortion is reduced to \( D = 2/\lambda^2 = \sigma_s^2 \), where \( \sigma_s^2 \) is the variance of a Laplacian distribution. A distortion model for a mixture-Laplacian distribution is easily constructed by linearly combining (13) with the corresponding distortion model for a mixture-Laplacian distribution is easily estimated using integration:

\[
\int_0^{\Delta-1} x^2 e^{bx} dx = e^{bx} \left( \frac{x^2}{b} - \frac{2x}{b^2} + \frac{2}{b^3} \right) \bigg|_0^{\Delta-1} = \xi,
\]

where \( \xi \) is given by (14). Next consider term \( \sum e^{bk\Delta} \) in (16) and notice that it is a geometric series with the following expansion:

\[
\sum_{k=0}^{N/\Delta} e^{bk\Delta} = \frac{1 - e^{b(N+\Delta)}}{1 - e^{b\Delta}} \approx \frac{1}{1 - e^{b\Delta}},
\]

where the last approximation holds since \( e^{b(N+\Delta)} \) is negligible and can be omitted for all practical values of \( N \) and \( b \). Multiplying (19) by \( 2\alpha \) and \( \xi \), we obtain (13). Notice that when \( \Delta = 1 \), (13) produces \( D = 0 \) and when \( \Delta = \infty \), the distortion is reduced to \( D = 2/\lambda^2 = \sigma_s^2 \), where \( \sigma_s^2 \) is the variance of a Laplacian distribution. A distortion model for a mixture-Laplacian distribution is easily constructed by linearly combining (13) with the corresponding probability \( p \) and \( 1-p \) as shown in (11). The result of applying model (13) to frame 0 in CIF Foreman is shown in Fig. 5 (left).

We extensively analyzed the performance of model (13) in other sequences and found that it was very accurate. Fig. 5 (right) compares the performance of (13) to that of the classical model (6) and UQ model (7) in FGS-coded CIF Coastguard. The error in the figure is computed for each frame in the PSNR domain and then averaged over all bitplanes. As the figure shows, (13) maintains the average error below 0.8 dB, while the errors in the other two methods average between 2 and 6 dB.

Note, however, that this form of averaging can be misleading since large errors in the last bitplane (where they do not matter due to high signal PSNR) may skew the result obtained from the other bitplanes. Thus, in Table II, we examine the average errors for each bitplane over the entire CIF Foreman sequence (similar results hold for Coastguard and Carphone). As the table shows, the PSNR error is quite small for all bitplanes except the last one where approximation (17) is most weak and results in the largest discrepancy between the model and the data. It is also worthwhile to note that a 1-dB error in a signal reconstructed at 56 dB is not noticeable, as well as that 0.15-dB errors in 30+ dB signals are relatively minor.

Finally note that (13) applies to any Laplacian source regardless of reconstruction points and whether the source contains FGS residue or base-layer DCT coefficients.

### Table II

**Estimation Accuracy of (13) in CIF Foreman**

<table>
<thead>
<tr>
<th>( \Delta )</th>
<th>Average ( D )</th>
<th>Average abs.error</th>
<th>Error in dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>81.5 (29.9 dB)</td>
<td>2.987</td>
<td>0.15</td>
</tr>
<tr>
<td>32</td>
<td>51.6 (31.2 dB)</td>
<td>1.768</td>
<td>0.15</td>
</tr>
<tr>
<td>16</td>
<td>23.1 (34.6 dB)</td>
<td>0.558</td>
<td>0.10</td>
</tr>
<tr>
<td>8</td>
<td>7.92 (39.2 dB)</td>
<td>0.239</td>
<td>0.13</td>
</tr>
<tr>
<td>4</td>
<td>2.16 (44.6 dB)</td>
<td>0.128</td>
<td>0.24</td>
</tr>
<tr>
<td>2</td>
<td>0.62 (49.8 dB)</td>
<td>0.039</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>0.08 (56.6 dB)</td>
<td>0.043</td>
<td>1.15</td>
</tr>
</tbody>
</table>

---

A. Preliminaries

In fine-grained scalable coders (e.g., FGS, PFGS), bitplane coding is applied to DCT residue in the enhancement layer to achieve high flexibility during transmission (i.e., the bitstream can be truncated at any codeword). Even though bitplane coding is more efficient than common run-level coding in the base layer [24], modeling the bitrate of bitplane-coded data is rather difficult since each bitplane has a different correlation model.

Recall that the traditional bitrate model (7) can be viewed as a linear function of \( z \) in the bitplane domain (i.e., a linear function of \( \log \Delta \)). While this linear approach may be acceptable for a high-level description of R-D properties of the source, in practice a more accurate model is often needed. Fig. 6 (left) illustrates that the traditional framework (7) is accurate only at very high bitrates (i.e., large \( z \)). Furthermore, as the figure shows, the straight line of the traditional model does not account for the non-linear shape of the curve in this particular frame. Since this mismatch is predominant in FGS-coded video, we seek an alternative explanation for the shape of the curve.

One possible way of modeling the run-length coding (RLE) structure of bitplane coding is to analyze the distribution of runs within each bitplane. This naturally leads to \( n \) distributions per frame, where \( n \) is the number of bitplanes. An example of this modeling is shown in Fig. 6 (right), which illustrates the histogram of run-length coefficients in frame 84 of CIF Foreman for two extreme cases of \( \Delta = 1 \) and \( \Delta = 32 \) (similar plots are shown in [22]). In Fig. 6 (right), both histograms can be modeled by exponential (geometric) distributions (i.e., straight lines on a log scale) with high accuracy if we ignore the all-zero blocks. This approach is fairly straightforward, but does require modeling many minor
The bitrate of each bitplane in scalable coding is 
real coded bits
traditional model
1.E-05
1.E-04
1.E-03
1.E-02
1.E-01
1.E+00
0 25 50 75 100
RLE coefficient
Delta=1
Delta=32
0
1
2
3
4
5
6
bitplane z
Rate R (bits/pixel)

Fig. 6. Actual FGS bitrate and that of the traditional model (7) in frame 0 of CIF Foreman (left). The distribution of RLE coefficients in frame 84 of CIF Foreman (right).

The main difficulty in a direct application of the above Markov modeling to FGS data is that (22) assumes that runs of 1s are also RLE-coded. However, in FGS bitplane coding, only runs of 0s are coded with RLE, and each occurrence of a 1 produces a special symbol that needs to be separately VLC coded. As a result, model (22) is accurate only for the first several bitplanes (which contain very few 1s) and then starts to significantly underestimate the actual bitrate (by as much as 20-30%).

To solve this problem, we next extend the original model (22) and derive Markov-based entropy that reflects the coding process of many embedded coders.

Lemma 3: The bitrate of each bitplane in scalable coding is given by a modified Markov model:

\[ H(z) = p_0H(X|S_1) + H(X|S_0), \]

where \( p_0, H(X|S_0), \) and \( H(X|S_1) \) are computed separately for each bitplane \( z \) using (20), (24), and (25), respectively.

Proof: Since there is no specific codeword for 1s in the assumed bitplane coding, we pursue a different approach for normalizing the entropy of each state as compared with the traditional approach in information theory. Instead of modeling the entropy of 1-runs and dividing it by the average length of a 1-run, we count the entropy of state \( S_1 \) as if it were a part of state \( S_0 \) and then divide both entropies by the length of the average zero-run. Thus, the average entropy is given by:

\[ H(z) = \frac{H_0 + H(X|S_1)}{r_0}, \]

where \( H_0 \) is the entropy of \( S_0 \) and \( r_0 \) is the expected length of a zero-run. Next notice that the probability to encounter a zero-run of length \( r \) is given by a geometric distribution \( P_0(r) = (1 - p_0)^{r-1}p_0 \) and that the entropy of the zero state is [29]:

\[ H_0 = - \sum_{r=1}^{\infty} P_0(r) \log_2 P_0(r) \]

\[ = - \left[ \log_2 p_0 + \left(1 - p_0 \right) \frac{\log_2 \left(1 - p_0\right)}{p_0} \right]. \]

Finally, it is easy to see that:

\[ r_0 = \sum_{r=1}^{\infty} r P_0(r) = 1/p_0. \]

Combining (25) and (28) in (27), we obtain the modified Markov model (26).

Examples in Fig. 8 shows the actual entropy rates for several frames in the CIF Foreman sequence, fitted with classical (22) and modified (26) Markov models. As the figure shows, the traditional approach does in fact underestimate the bitrate of the last bitplane by as much as 30%, while the modified model is capable of tracking rate \( R \) over the entire range of bitplanes \( z \). Note that these results directly apply only to the rates of individual bitplanes \( z \). Thus, if the server transmits all bitplanes up to and including bitplane \( z \), the cumulative rate \( R(z) \) is the summation of the individual bitplane rates:

\[ R(z) = \sum_{k=1}^{z} H(k), \]
where \( z = n - \log_2 \Delta \) and \( n \) is the total number of bitplanes in the frame. Fig. 9 and Fig. 10 show the remarkable accuracy of the final result (30) in modeling accumulative rate \( R(z) \).

Although it requires two estimated probabilities \( (p_0, p_1) \) per bitplane\(^4\), this approach is highly accurate. Combined with the distortion model (13), the result in (30) allows the construction of accurate R-D curves only based on the statistical properties of DCT residue.

C. \( \rho \)-domain R-D Model

Although model (30) is quite accurate, it requires a non-trivial effort in obtaining transition probabilities for each bitplane and a rather large set of configuration parameters (10-14 parameters per frame) that may be undesirable in real-life streaming situations. Therefore, we next investigate an alternative R-D model that requires much fewer parameters than the Markov model.

Lemma 4. For Laplacian sources, distortion \( D(R) \) can be expressed in closed form as:

\[
D = \sigma_x^2 - (c_1 \log^2 R + c_2 \log R + c_3)R,
\]

where constants \( c_1 - c_4 \) only depend on the shape of the distribution \( \lambda \) and \( \rho \)-domain constant \( \gamma \) [12].

Proof: First recall that He et al. [12] demonstrated in numerous simulations that in a variety of image and video coding methods, rate \( R(z) \) was proportional to the percentage of non-zero coefficients \( 1 - \rho \) in the source data:

\[
R(z) = \gamma(1 - \rho),
\]

where \( \gamma \) is a constant. While we do not offer a deeper analytical treatment of (32) at this time, we utilize this empirical fact in our subsequent derivations, especially since this model holds very well for scalable coders (not shown here for brevity, but verified in simulations). Next notice that for Laplacian distributed sources, the percentage of non-zero coefficients is:

\[
1 - \rho = 1 - 2\int_0^{\Delta} \frac{\lambda}{2} e^{-\lambda x^2} dx = e^{-\lambda \Delta},
\]

where \( \lambda \) is the shape parameter of the Laplacian distribution and \( \Delta \) is the quantization step. Inserting (32) into (33), we express \( \Delta \) in terms of rate \( R \):

\[
\Delta = -\frac{1}{\lambda} \log \frac{R}{\gamma},
\]

Combining this result (34) with our earlier distortion model (13), we have:

\[
D(\Delta) \approx \frac{\lambda \xi}{1 - R/\gamma},
\]

where \( \xi \) is:

\[
\xi = \frac{2}{\lambda^3} - \frac{\epsilon^2 R}{\lambda^3 \gamma} \left( (\log R - \log \gamma + \lambda - 1)^2 + 1 \right).
\]

Expanding (36) and combining it with (35), we notice that:

\[
D = \begin{cases} 
\frac{2}{\lambda^3} = \sigma_x^2, & R = 0 \\
0, & R \geq e^{-\lambda \gamma} 
\end{cases}
\]

where \( \sigma_x^2 \) is the variance of the source. This observation makes perfect sense since distortion \( D \) should not be larger than \( \sigma_x^2 \) [5] and should equal 0 when \( R = e^{-\lambda \gamma} \) (i.e., the quantization step \( \Delta = 1 \) and there is no loss of information). After absorbing the various constants and neglecting small terms, we have the desired result in (31).

Estimation of \( \gamma \) for FGS sources is very simple. Once the FGS layer is coded, the number of bits \( R(z) \) in each bitplane can be easily obtained by scanning the FGS layer for bitplane start codes (whose location can also be saved during encoding). Computing the percentage of zeros \( \rho_z \) in each bitplane directly from the DCT residue, the encoder can build the curve \((1 - \rho_z, R(z))\) and estimate its linear slope \( \gamma \). Simulation results in Fig. 11 show that model (31) outperforms traditional R-D models and maintains high accuracy in a variety of FGS-coded video frames.
The result in (31) shows that the R-D curve of Laplacian sources is both complex and highly non-linear in both the MSE and PSNR domains. Nevertheless, this model provides valuable insight into the coding process and suggests the shape of the resulting R-D curve. For practical purposes, this model is still rather complex and hard to use for R-D analysis due to the numerous non-linear terms in (36). Thus, we examine an even simpler operational model in the next section and use it for quality control in the subsequent parts of the paper.

VI. SQUARE-ROOT R-D MODEL

A. Simple Quality (PSNR) Model

Notice that the previously derived distortion model is too complicated for further analytical manipulation. In the following discussion, we convert $D$ into the PSNR domain and reduce it to a simpler formula through a series of approximations. Taking the logarithm of (13), omitting insignificant terms, and grouping constants, we obtain:

$$\log D(\Delta) \approx c_1 + e^{b\Delta} + b\Delta + \log(c_2\Delta^2 + c_3\Delta + c_4).$$

for some constants $c_1, \ldots, c_4$. In the working range of most video coders, $\Delta$ is no more than 128 and the number of bitplanes usually does not exceed 7. In this limited range, a number of approximations hold: $\log(x^2 + x + c) \approx a \log^2 x + b \log x + c$ and $x + e^{ax} \approx a \log^2 x + b \log x + c$, for some constants $a\sim c$. Then, (38) can be further simplified to:

$$\log D(\Delta) \approx c_1 \log^2 \Delta + c_2 \log \Delta + c_3.$$

(39)

Since $\Delta = 2^{n-z}$, (39) shows that PSNR curves of this approximation are quadratic polynomials of the bitplane number $z$:

$$PSNR(z) \approx g_1z^2 + g_2z + g_3,$$

for some constants $g_1, \ldots, g_3$. This expression is very useful since polynomials are easy functions to work with and smoothly generalize the linear model of the traditional framework where $g_1$ equals zero.

To verify this approximation, we conducted a series of tests by fitting the simplified model (40) to the PSNR calculated from the original model (13) and found them to be an almost perfect match. The quality of the fit is illustrated on two different Laplacian distributions in Fig. 11. The left side of the figure shows a low-variance (high $\lambda$) case and the right side of the figure shows a high-variance (small $\lambda$) case; both matched the quadratic model (40) with very high accuracy.

B. Simple Bitrate Model

We first need the following supplementary result.

**Lemma 5**: Function $R(z)/\gamma$ for $z \in [1, n]$ is monotonically increasing, changes convexity no more than once, and remains in $[0, 1]$ for all bitplanes $z$.

**Proof**: Combining (32) with (33) and keeping in mind that $\Delta = 2^{n-z}$, we have:

$$\psi(z) = \frac{R(z)}{\gamma} = e^{-\lambda 2^{n-z}} < 1.$$  
(41)

Taking the first two derivatives of (41), we have:

$$\psi'(z) = \lambda 2^{n-z} \log(2) \psi(z) > 0,$$

(42)

$$\psi''(z) = \lambda \log(2) [-2^{n-z} \log 2 \psi(z) + 2^{n-z} \psi'(z)].$$

(43)

Analysis of (43) shows three important points: (a) for $\lambda \geq 1$, the function $\psi$ remains strictly convex in the entire interval, (b) for $\lambda \leq 2^{1-n}$, the function remains strictly concave, and (c) for the remaining values of $\lambda$, there is exactly one point $z = n + \log_2 \lambda$, in which the function changes convexity.

Using the theory of coconvex/comonotone approximation [21], an accurate polynomial approximation of $R(z)$ would require a cubic curve to match the possible change in convexity of the curve (the rest of the error is small since (41) exhibits a good degree of smoothness). However, since working with cubic polynomials is still rather complex (e.g., for realtime rate-control applications), we apply a quadratic approximation to $R(z)$ in the $z$-domain and reduce (41) to:

$$R(z) = a_1z^2 + a_2z + a_3,$$

(44)

where constants $a_1, \ldots, a_3$ can be estimated from empirical data. To better understand this operational model, we conducted numerous experiments and found that while cubic polynomials were a very good match to $R(z)$, quadratic functions also performed well. Fig. 13 shows one such example for two frames of CIF Foreman, as well as a linear fit derived from model (7).

C. SQRT Model

We next combine our proposed bitrate result in (44) with the earlier distortion model in (40) to obtain a final usable R-D model. After inverting the polynomial in (44), inserting $z(R)$ into (40), and dropping insignificant terms, we obtain the model that we call Square Root (SQRT):

$$PSNR(R) = AR + B\sqrt{R} + C,$$

(45)
where constants $A$ and $B$ are estimated from at least two (R,D) samples, and $C = 10 \log_{10}(255^2/\sigma_x^2)$ for uncorrelated (or weakly correlated) sources such as those in FGS coders. Parameter $A$ and $B$ are strongly negative-correlated (e.g., the 0-lap cross-correlation coefficient between these two parameters is -0.99 in the CIF Foreman sequence).

We next revisit two “difficult” PSNR curves shown earlier in Fig. 3, in which even a quadratic polynomial of $R$ was unable to follow the curve. Fig. 14 shows the new result for the SQRT model (45) and demonstrates a much better fit than was possible before.

To better understand the estimation accuracy of the different models discussed so far, we compare the SQRT model (45), Chiang’s model (9), the UQ model (7), and classical model (6) in various video sequences. Fig. 15 and Fig. 16 show the average absolute error between the actual R-D curve in the PSNR domain and each of the models in several FGS-coded sequences. For example, in the FGS-coded Foreman sequence, the error in SQRT averages 0.25 dB, while it stays as high as 2-8 dB in the other three models. Finally, note that we tested (45) in numerous other sequences, as well as at different base-layer bitrates, and found it to significantly outperform traditional models, which often required estimation of the same number of parameters.

We also examined the accuracy of SQRT in PFGS. Recall that PFGS uses prediction in the enhancement layer to achieve better compression in sequences with high degrees of temporal correlation. Assuming that all predicted bits are transmitted to the client, our derivations and models are applicable to PFGS. Fig. 17 shows that model (45) outperforms the traditional R-D model in PFGS-coded sequences. The figure also shows that the UQ model and Chiang’s model have large error variation in these sequences, which happens because PFGS not only uses the enhancement layer for prediction but also for reconstruction, which is beyond the range of the UQ model and Chiang’s model.

We conclude this section by noting that (45) takes the following simple shape in the distortion domain:

$$D = e^{2aR + b \sqrt{R}}$$  \hspace{1cm} (46)

where $a < 0$, $b$ are constants and and $c$ is proportional to the source variance. This is a generalization of the traditional R-D function $D = e^{-2R}$, in which $b = 0$.

VII. R-D BASED QUALITY CONTROL IN INTERNET STREAMING

In streaming applications, fluctuating visual quality is often unpleasant to the humans, who are normally used to relatively constant quality found in broadcast TV, VCR, and DVD programming [42], [43]. However, due to the inherent nature of current video coding schemes, the base layer usually suffers from substantial quality fluctuation, as shown in Fig. 18 (left) for Foreman CIF (note a 6-dB drop in quality within just a 10-second fragment).

Therefore, one of the goals of streaming servers is often to select such quantities of the enhancement layer that provide constant PSNR quality at the receiver. Furthermore, by rescaling the enhancement layer according to its R-D curve, the server can not only provide low fluctuation of video quality, but also match the available bandwidth in the network. The latter goal is achieved by coupling rate-scaling decisions with congestion control. In what follows in this section, we first describe very simple R-D based constant-quality streaming
and then examine asymptotically stable congestion control methods that provide a foundation for oscillation-free transport of video over the Internet.

### A. Quality Control Algorithm

As we mentioned earlier, rate control is one popular application of R-D models. The main question here is how to scale the FGS layer to both match the available bandwidth $R_T$ (total bits allowed for the entire sequence) and achieve certain constant quality $D$ after decoding. We illustrate the solution to this problem using Fig. 18 (right) and a simple sequence consisting of two frames. First, the server inverts the result in (45) or (46) and obtains two $R(D)$ curves (one for each frame). Second, it generates the combined rate curve $R_1(D) + R_2(D)$, which shows the amount of total bits required to achieve constant $D$ in both frames. Knowing $R_T$, the combined curve needs to be inverted one more time to obtain the value of $D_T$ that provides the required total bitrate $R_T$. The size of individual frames is given by $R_1(D_T)$ and $R_2(D_T)$ as the final step.

For longer sequences, the server adds the R-D curves of all frames and obtains a combined function $F(D)$, which is constrained by $R_T$:

$$F(D_T) = \sum_{i=t}^{N} R_i(D_T) = R_T,$$

where $R_i(D)$ is the R-D function of frame $i$, $N$ is the number of frames in the sequence, and $t$ the frame at which the server decides to change its rate $R_T$ in response to congestion signals. Partial summation in (47) is important since congestion control often changes its rate in the middle of actual streaming and (47) needs to be recomputed every time such a change is encountered. Finding the root of (47) involves inverting $F(D)$ and evaluating

$$D_T = F^{-1}(R_T).$$

Once $D_T$ is known, each enhancement layer frame $i$ is scaled to $R_i(D_T)$ and then transmitted to the receiver. Even though the new R-D framework does not lead to a closed-form solution for $F^{-1}$, each of the individual curves can be generated with high accuracy using only a 3-point interpolation and the resulting function $F(D)$ can be computed (and then inverted) very efficiently.

In Fig. 19, we illustrate this simple rate control algorithm applied to the SQRT R-D model assuming that the channel capacity is fixed (variable channel rates are studied in the next section). The figure shows simulation results using Foreman CIF with 768 kb/s available in the network for the enhancement layer in comparison with two other rate-control methods – those proposed in the JPEG2000 [16] image coding standard and in Wang et al. [35]. Experimental results show that the new R-D framework can be successfully used to both dramatically reduce undesirable quality fluctuation during streaming and to relieve the server from expensive interpolation. The variance in PSNR between adjacent frames in the SQRT curve is only 0.04 dB in Fig. 19 (left) and 0.004 dB in Fig. 19 (right).

Many constant quality control approaches in related work stop after solving the problem for CBR channels [35], [42], [43]. We, on the other hand, find that the neither the exact method of scaling the enhancement layer (this section), nor the underlying R-D model (the previous section) are very important if the application relies on any of the wide variety of AIMD-style congestion control methods. Hence, we feel that with goals of constant-quality streaming, it becomes more important to continue the research into the area of smooth congestion control, which is a pre-requisite to actual implementation of any of these methods. Unfortunately, the current Internet does not provide an environment where smooth (asymptotically stable) sending rates can be easily achieved; however, there are promising classes of congestion controllers for the future Internet than may fulfill these requirements. One such class is studied next.
B. Congestion Control Overview

There are many challenges facing Internet streaming applications, all of which stem from the lack of quality-of-service (QoS) guarantees in the transport layer. One of the primary impediments to high-quality delivery of real-time video to the end user is the variable channel bandwidth. Notice that even though end-to-end paths often experience relatively stationary conditions (in terms of the number of competing flows, average long-term packet loss, etc.), current congestion control methods built on top of a variety of TCP-friendly schemes cannot asymptotically converge (from a control-theoretic point of view) to a single stationary rate or provide a smooth “virtual” channel to the video application.

Recently, a major effort has been dedicated to developing smoother congestion control methods for multimedia streaming (e.g., TFRC [9] and binomial algorithms [1]). Nevertheless, these newly-developed methods are not asymptotically stable, nor do they have any stationary points in the feasible operating range of a typical application [40]. Note that unless a video application can employ a stable congestion controller, any attempts to provide constant-quality streaming will be moot.

In this section, we study continuous-feedback congestion controllers proposed by Kelly et al. [18] and investigate whether their performance provides the necessary foundation for achieving the goals of this paper.

C. Kelly Controls

Recall that TCP and classical binary-feedback methods (such as AIMD and binomial algorithms) rely on packet loss in order to increase or decrease their rates. Since the decision about changing the current rate is binary, we can summarize their control functions as following:

\[
\frac{dr}{dt} = (1 - sgn(p))F(r) - sgn(p)G(r),
\]

where \( r(t) \) is the rate at time \( t \), \( p(t) \) is packet loss, \( F(r) \) is the increase function, and \( G(r) \) is the decrease function. Notice that with a reasonable choice of functions \( F \) and \( G \), the right side of (49) does not have roots, which means that the equation does not have stationary points. Since (50) cannot be stabilized, it must oscillate or diverge. It is easy to show that under certain mild conditions on \( F(r) \) and \( G(r) \), (51) oscillates around the equilibrium (equal-share) rate. The amount of oscillations depends on the choice of \( F(r) \) and \( G(r) \) and typically leads to a trade-off between the size of oscillations and the rate of response to congestion signals. Thus, controls that produce small oscillations are usually susceptible to more packet loss due to their reluctance to back off during congestion.

What is interesting about binary-feedback methods is that they typically do not possess any methods that can force the oscillations to asymptotically decay to zero, even under stationary cross-traffic conditions. Therefore, we seek alternative methods that provide this functionality and are provably stable under both immediate and delayed feedback. One such alternative is given by Kelly’s congestion control framework called proportional fairness [18]:

\[
\frac{dr}{dt} = r(\alpha U'(r) - \beta \sum_{i \in P} p_i),
\]

where \( U(r) = \log r \) is the utility function of the end user, \( \alpha > 0 \) and \( \beta > 0 \) are constants, and \( p_i \) is the price that the flow pays for using resource (router) \( l \) along the end-to-end path \( P \). Kelly’s controls have received significant attention in the theoretical networking community [15], [18], [19], [26]; however, their application in real networks or streaming applications has been limited.

Notice several clarifications of the original framework (50) that are necessary to make this controller practical. First, it is common to use packet loss as the continuous feedback (instead of the price) simply because the current Internet is still best-effort and prices are a meaningless metric for individual routers. Second, instead of summing up the packet loss experienced by all routers of an end-to-end path, it sometimes makes more sense to use the maximum packet loss among these routers in order to match the rate of the application to the bandwidth of the slowest link in the path:

\[
p(t) = \max_{i \in P} p_i. \tag{51}
\]

Expanding (50) using a single feedback \( p(t) \) of the most-congested resource and converting the system into the discrete domain, we have a more application-friendly version of the controller:

\[
r_i(t) = r_i(t - D_i) + \alpha - \beta r_i(t - D_i)p(t - D_i^{-}), \tag{52}
\]

where \( i \) is the flow number, \( D_i \) is its round-trip delay, and \( D_i^{-} \) is the backward feedback delay from router \( l \) to user \( i \). Note that this version of Kelly controls includes novel maxmin changes to the feedback and an extra delay applied to the additive term \( r_i(t - D_i) \) in (52). Full analysis of this framework is beyond the scope of this paper, but the following important result is available in our recent work.

Lemma 6: Discrete controller (51)-(52) is asymptotically stable and fair regardless of round-trip delays \( D_i \), the exact shape of packet loss \( p(t) \), or feedback delays \( D_i^{-} \) as long as \( 0 < \beta < 2 \).

Proof: See [41].

Our final issue to address is the shape of packet loss \( p(t) \). While (51)-(52) can operate in the end-to-end context where \( p(t) \) is estimated by the receiver, we find that involvement of AQM (Active Queue Management) significantly improves the performance of this controller. Thus, each router performs a very simple operation of counting the total arriving traffic into each queue, dividing the result by the fixed duration of the control interval, and inserting feedback \( p_i(t) \) into packets passing through the queue:

\[
p_i(t) = \frac{\sum_{i \in S_l} r_i(t) - C_l}{\sum_{i \in S_l} r_i(t)}, \tag{53}
\]

where \( S_l \) is the set of flows passing through resource \( l \) and \( C_l \) is the speed of the resource (i.e., its outgoing bandwidth). Each router needs to maintain one variable with the total number
of bytes placed in the outgoing buffer during the last $T$ time units. At the end of each interval, this counter is divided by $T$ to obtain an estimate of $\sum_{i \in S} r_i(t)$, which is then used to calculate $p_t$ using (53). The new value of $p_t$ is inserted into each passing packet as long as the corresponding $p_{t-1}$ contained in the packet is lower than the value computed by this router. Notice that the router does not need to count the number of flows or estimate their individual rates $r_i$. This means that the feedback is based on the aggregate flow rate $\sum_{i \in S} r_i(t)$ rather than on individual flow rates. This in general increases the scalability of these AQM functions inside each router. For additional implementation discussion, see [17].

It is also possible to demonstrate that the convergence rate of Kelly controls is at least exponential, which makes this framework appealing for future very high-speed networks.

**Lemma 7:** Under AQM feedback in (53), controller (51)-(52) reaches link utilization exponentially fast.

**Proof:** See [41].

The result of this lemma is illustrated in Fig. 20, in which $\beta = 0.5$ and $\alpha = 10$ kb/s. The figure shows that it takes 8 steps for a single-flow to fill a 1.5 mb/s T1 bottleneck and it takes only 16 steps for the same flow to fill a 10 gb/s link. Note that both flows reach within 5% of $C$ in just 6 steps.

**D. SQRT Quality Control in VBR Networks**

In this section, we examine the PSNR quality curves when the target rate $R_T$ is not known a-priori, but is rather supplied by real-time congestion control. We obtained the traces of $r(t)$ from ns2 simulations and then applied them to the video scaling algorithm offline. We should point out that one limitation of this approach is that we did not take into account the effect of lost packets during the simulation on the quality of the stream. This is reasonable in streaming scenarios where the application protects its packets by FEC or some form of retransmission. Since in Kelly controls, the amount of packet loss $p^*$ in the steady state is fixed and known to the end flow once it reaches the equilibrium [41], it becomes easy to send enough FEC to cover the exact amount of lost data.

To set a baseline example, in Fig. 21 (left), we compare the AIMD (1, 0.5) control with modified framework (52) using PSNR quality curves. In this simulation, a single flow is run over a bottleneck resource of capacity $C = 1$ mb/s (the round-trip delay is 100 ms). As the figure shows, both controls at first follow the PSNR of the base layer since there is not enough discovered bandwidth to send any FGS data. Once this stage is passed, both controls achieve high PSNR; however, the difference is that AIMD backs off by half upon every packet loss, while Kelly controls eventually stabilize at a fixed rate. Rate fluctuation in AIMD results in periodic jumps (sometimes as high as 4 dB) throughout the entire sequence.

Fig. 21 (right) shows another scenario where two Kelly flows are sharing the same bottleneck link $C$ under identical 100-ms round-trip delays. Flow1 in the figure is started with $r_1(0) = C$ and flow2 is started with its base-layer bandwidth. As seen in the figure, the two flows converge to a fair allocation at approximately $t = 3$ seconds and then follow the same flat quality curve.

The next issue to examine is whether different round-trip delays $D$ have any effect on fairness. Fig. 22 (left) shows a scenario in which two flows with different RTTs start in the same unfair states as before. The corresponding delays are 400 and 100 ms; however, this has little effect on the resulting fairness as both flows stabilize at 34.5 dB around $t = 7$ seconds.

We also examine the effect of random feedback delays on our quality-control framework, in which the round-trip delay is uniformly distributed between 100 and 400 ms and the initial states are as before. Fig. 22 (right) shows that although the convergence is somewhat slower than in the previous examples ($t = 8$ seconds), both flows manage to provide a stable quality after the convergence. This confirms our earlier result regarding stability of (51)-(52) under arbitrary delays.

In summary, Kelly controls converge to equilibrium without oscillation and then stay there as long as the number of flows at the bottleneck remains fixed. When new flows join or leave, the transition between fair (equilibrium) points is monotonic in
most situations. This provides a nice foundation for video-on-demand and other entertainment-oriented video services where each flow is long-lived and can take full advantage of this smooth congestion control framework.

VIII. CONCLUSION

This paper modeled the statistical properties of DCT residue and presented a detailed analysis of bitrate and distortion of fine scalable coders. After obtaining an efficient operational R-D model, we applied it to Internet streaming for quality control purposes. As demonstrated in simulation, our R-D quality control algorithm works well in achieving constant quality not only in CBR networks, but also in VBR channels coupled with congestion control. To overcome the limitations of TCP-friendly methods, we used modified Kelly controls and showed that they can achieve stable sending rates in practical network environments and provide an appealing framework for future high-speed AQM-enabled networks.

REFERENCES