Rate-Distortion Optimized Client Side Rate Control for Adaptive Media Streaming

Sanjeev Mehrotra, Weidong Zhao

Microsoft Research, One Microsoft Way, Redmond, WA 98052
{sanjeevm, weidongz}@microsoft.com

***Abstract*—Media streaming over unreliable networks such as the Internet is growing in popularity, but presents unique challenges when trying to get the user experience to be on par with classical mediums such as cable television. These networks have variable network conditions which not only vary between a set of points in the network, but also change over time. In this paper, we present a rate-distortion (R-D) optimized algorithm for adapting the bitrate of streaming media from a chunked encoding, where each chunk is available at multiple bitrates (or quality levels) or is scalable coded. The optimization takes into account the desired startup latency, the desired client buffer size, the current client buffer size, and the estimated network bandwidth. The problem is formulated as a distortion minimization problem subject to multiple rate constraints. For video content, the solution gives a gain of up to 3-4dB in PSNR in difficult portions of the video when compared to commonly used adaptation techniques and can achieve an arbitrarily small desired startup latency.**

I. Introduction

Video on demand (VOD) and live video over the Internet is growing in popularity as evidenced by the large number of websites offering television (TV) content and movies. Among the common approaches for delivery are streaming and download-and-play (or progressive download). Although streaming is meant to be close to watching TV – where the content starts to play soon after the user wants – until recently, most of it used to be much worse than classical mediums. Large buffering delays when initially starting and when seeking, as well as glitches due to insufficient bandwidth were fairly common. To avoid glitching and rebuffering during playback, many sites reverted to download-and-play or progressive download techniques even if the content bitrate was less than the network bandwidth, resulting in a large startup latency. Although modern streaming solutions have alleviated some of these issues, there is still room for significant improvement.

Initial streaming solutions used to use a single constant bitrate (CBR) coding. CBR encoding assumes a constant network bandwidth, and thus uses startup latency, buffer size (to handle content variability), and bitrate as constraints when encoding. Given these constraints, the encoder attempts to maximize quality. If these constraints match with actual constraints during playback, the CBR encoding works well. However, in general the constraints don't hold since bandwidth is variable. Even if the bandwidth does not vary, the startup latency constraint assumes that there is a place where content starts. This in general is not true since the user may seek to a different locations. Thus, even if an encoder has optimized for the startup latency to be small when playing from the beginning of the file – which it usually doesn't do – seeking will still result in a buffering delay. The main cause of poor quality when streaming is this constraint mismatch between the encoding and playback side.

To try to solve the constraint mismatch problem, solutions were proposed to use streams encoded at multiple bitrates (MBR) or scalable coding so that one of the bitrates can match the current playback conditions. As the network bandwidth changes, the client switches to different bitrate streams [1]. As a followup to this, other works have proposed better bitrate adaptation strategies to optimize for startup latency as well as allow the client buffer fullness to follow some desired target schedule [2], [3]. However, the bitrate adaptation techniques developed so far have only attempted to satisfy the rate constraints (network bandwidth, desired startup latency, desired buffer size). In particular, they have not used a rate distortion optimized framework for actually minimizing the distortion subject to these constraints.

Initial MBR solutions also had other issues. Let's use video as an example. To maximize compression efficiency, encoders use fairly large GOP (Group Of Pictures) sizes which limits the flexibility in bitrate adaptation as the client has to wait for the next I-frame before switching. Also, each bitrate is coded independently, so although switching to one bitrate may be allowed at a given point, switching to another may not as it may not be the start of a new GOP in the stream. Furthermore, since each stream was essentially a separate CBR encoding, the buffer that a single CBR encoding thinks it has at a given point in time when making encoder decisions may not be the actual buffer, and thus if the switch to another bitrate is not made early enough, it will still result in a rebuffering delay.

To solve these issues, modern streaming solutions such as Move Networks [4], and adaptive streaming using Adobe Flash and Microsoft's Silverlight [5] have started using small GOP sizes and aligned GOPs across the bitrates to make it easier to switch from one bitrate to another. This has removed the cross-GOP dependency on the decoder side. To remove cross GOP dependency on the encoder side, instead of using a CBR encoding of the entire file (which results in buffer state dependencies across GOPs), they independently code each of the GOPs using a bitrate constrained variable bitrate (VBR) encoding. Thus an entire GOP encoding has to be

MMSP'09, October 5-7, 2009, Rio de Janeiro, Brazil.
978-1-4244-4464-9/09/$25.00 ©2009 IEEE.

downloaded prior to starting playback of a GOP. Although breaking the cross GOP dependency results in suboptimal compression efficiency, it makes bitrate adaptation easier and minimizes buffering (latency) and glitches. The suboptimal compression efficiency is tolerated as network bandwidths are higher. Although these modern streaming solutions provide an improved user experience, there is still room for improvement in the following areas which this paper addresses.

- These solutions employ a fixed number of fixed rate encodings for each GOP. For example, each GOP may be 60 frames and available at 0.5, 1, and 2Mbps regardless of the content in the GOP.
- They use a pure rate based bitrate adaptation. For example, if the network bandwidth is found to be 1Mbps, then once the buffer is of a desired size, they will download the 1Mbps encoding for all chunks. Such a client strategy is highly suboptimal since some GOPs are much easier to code than others.
- The minimum startup latency is lower bound by the time it takes to download a chunk at the lowest rate.

This paper does not propose an alternative transmission protocol. The underlying transmission still uses something like TCP to handle congestion and flow control as it already works well for media streaming (especially once a buffer has been built).

In Sec. II, we propose a streaming solution which uses a modified version of the flexible coding presented above and dynamically adapts the content bitrate to satisfy the client's varying conditions. However, unlike previous work, the adaptation is formulated as a distortion minimization problem subject to multiple rate constraints (derived in Sec. II-A) which are a function of the current client buffer fullness, the desired client buffer fullness, estimated network conditions, the current time, and the playback deadline for portions of the content. Initially, the current time can be set to an arbitrary value to satisfy a desired startup latency. A solution to the optimization problem is presented in Sec. II-B. It gives an optimal solution if the content is encoded using a truly scalable encoding where the bitrate can be changed to any arbitrary bitrate. It gives an approximately optimal solution if there is a coarser sampling of bitrates or quality levels. Minimizing distortion subject to multiple rate constraints has been studied before in several papers [6], [7], [8].

In Sec. II-C, we also show how having additional encodings with additional constraints can be used to reduce the minimum startup latency to arbitrary values. By only using these encodings at startup or after a seek, they can be used to reduce latency without affecting the overall quality. They can also easily be incorporated into the rate distortion minimization framework as they only modify some of the rate constraints.

II. Rate Distortion Optimized Streaming

The media is divided into *chunks* so that it is seamless to start playback or switch from one bitrate to another at the chunk boundaries, which means that each chunk is coded independently. A chunk is a small segment of audio or video and can be variable in duration. Typically, for video, each chunk can be a closed GOP of certain duration. Each chunk is coded using either a scalable coding providing a very fine grained scalability down to the bit, or coarser scalability which provides a number of layers for each chunk (base layer plus enhancement layers). Alternatively, each chunk is coded at multiple quality levels or rates independently (MBR).

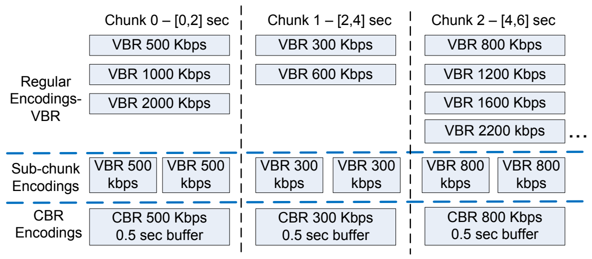

Fig. 1. Various encodings available for chunks. Either sub-chunk encodings or CBR encodings can be used to further reduce startup latency.

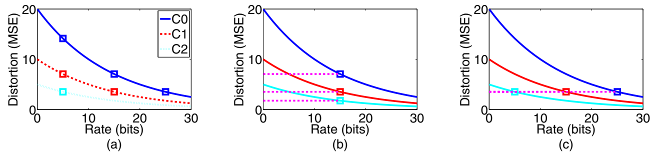

Fig. 2. Rate-Distortion curves for three chunks (C0, C1, C2). Available encodings are shown as squares in (a); (b) shows points achieved with a rate based only adaptation; and (c) shows points achieved with an R-D optimized adaptation.

Instead of coding all chunks at a fixed number of rates with the rates being the same for all chunks as is commonly done, we propose to code each chunk so that a reasonable sampling of the rate-distortion (R-D) curve is achieved as shown in Fig. 2(a). In the figure, we show the R-D curve for three chunks (C0, C1, C2) with the available encodings shown as squares on the curve. For the chunk which is most difficult to encode (C0), we can have three encodings (5, 15, and 25 bits), and for the chunk which is the simplest to encode (C2), we can have just one encoding (5 bits).

The encodings for each chunk can correspond to bitrate constrained VBR encodings at various rates as shown in Fig. 1. VBR is used to maintain constant quality through the duration of each chunk. Bitrate constrained VBR coding can be used to get a point with a desired rate on the R-D curve, and quality based VBR coding can be used to get a point with a desired distortion on the R-D curve. The selection of which encodings to obtain on the R-D curve is not covered here.

The scalable or MBR coded media is stored on a server along with an index file which contains information regarding chunk boundaries, chunk durations, and rate distortion points which are available for each chunk. This file is a small overhead when compared to the data rate. The index file can be downloaded in entirety prior to playback, or incrementally as the content plays. The current time is initialized based upon the desired startup latency. Then, at any given time, the client based upon its network bandwidth estimate and current buffer situation, computes how much additional information to download for chunks which are yet to be played and initiates the next download. Multiple simultaneous downloads can be used to improve throughput. After each download is complete,

TABLE I
NOTATION

N:	Number of chunks the media is divided into.
n:	Index of the chunk, $n = 0, 1, \ldots, N-1$.
$p[n]$:	Playback time of the first sample in chunk n.
$d[n]$:	Duration of chunk n.
$L[n]$:	The number of layers chunk n is encoded into, with the first layer being layer 1.
$F[t]$:	Playback time of the first chunk with playback time larger than t, $F[t] = p[\min\{i : p[i] > t\}]$.
$m[t,n]$:	$m[t,n] \in \{0, 1, \ldots, L[n]\}$ is the number of layers downloaded for chunk n at time t, the layers being downloaded sequentially from the first layer.
$Z[t]$:	Playback time of the first chunk which has no layers downloaded, that is $Z[t] = p[\min\{i : m[t,i] = 0\}]$.
$B[t]$:	The buffer at time t.
$R[n,l]$:	For chunk n, the rate needed to code all layers up to and including l.
$D[n,l]$:	The distortion achieved with layers up to and including l.
$\Delta R[n,l]$:	Additional rate needed when going from layer $l-1$ to l, $\Delta R[n,l] = R[n,l] - R[n,l-1]$.
$\Delta D[n,l]$:	Change in distortion (a negative quantity) going from layer $l-1$ to l. $D[n,0] = \infty$ or some other large value (such as the variance of the chunk).
$S[n,l]$:	The rate-distortion slope which is the decrease in distortion divided by the increase in rate when going from layer $l-1$ to layer l given by $S[n,l] = -\Delta D[n,l]/\Delta R[n,l]$.
κ:	Desired startup latency.
$T[n]$:	Time spent on downloading information for chunk n.

the bandwidth is re-estimated, and the process repeats. For scalable coded media, the client can continue to update chunks with additional enhancement layers so long as the playback deadline for the chunk has not passed (however, for MBR the R-D curve may no longer be convex). A seek essentially resets the process. We assume all downloads are sequential.

Performing the optimization at the client-side is beneficial because it reduces the server load and need for intelligence. Also, no additional server capability is needed beyond a simple web or file server, making it easy to deploy such a solution and take advantage of existing content distribution networks (CDN) and peer-to-peer networks.

When adapting the bitrate, the goal of the client is to minimize the total distortion for some chunks into the future, for a given distortion metric. For complexity and delay considerations, we don't have to look at all the chunks when optimizing. Existing client side adaptation strategies in use today attempt to match the bitrate of the chunk with that of the network. For example, in Fig. 2, if the bitrate of the network is found to be 15 (bits/chunk), then a simple strategy would download all three chunks at the same bitrate and achieve the rate-distortion points (shown by the squares) in Fig. 2(b). This would not only result in a suboptimal total distortion but also a wide range of qualities across the three chunks. On the other hand, a rate-distortion optimized strategy as shown in Fig. 2(c) would find points operating at the same slope on the R-D curve, which for this case would result in all chunks at the same quality (distortion) level. R-D optimization helps the most when there is large variability over the R-D curves for the chunks, which is likely to happen with small chunks.

In Table I, we go over notation used in the paper. The units of all time and duration quantities can either be in samples or seconds. The playback time of chunk n is given by $p[n] = p[0] + \sum_{i=0}^{n-1} d[n]$, that is $p[1] = p[0] + d[0]$, $p[2] = p[0] + d[0] + d[1]$, and so on. The buffer at time t is defined to be the time up to which the client can playback content without glitching, that is

$$B[t] = p[Z[t]] - t. \tag{1}$$

Some of this notation (playback time, duration, buffer size, time spent downloading) is shown in the timeline in Fig. 3.

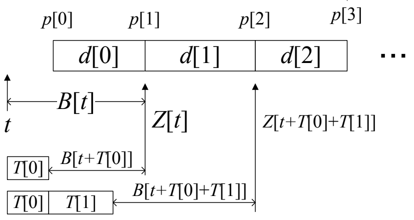

Fig. 3. Timeline showing chunks 0, 1, and 2, and buffer evolution as information is downloaded for chunks 0 and 1 taking time $T[0]$ and $T[1]$.

Initially, we set $m[t,n] = 0$ for all n since nothing has been downloaded. We also set the initial time to be $t = p[0] - \kappa$, that is the playback time of the first chunk minus the desired startup latency. This gives $F[t] = 0$, $Z[t] = 0$, and an initial buffer of $B[t] = \kappa$. Without loss of generality, for simplicity, in the remainder of the discussion, assume $F[t] = 0$ ($t < p[0]$), and thus the first chunk considered in the optimization is 0. Also, assume that N is the number of chunks considered in the optimization, which may be less than the total number of chunks in the media file due to complexity or delay concerns.

For MBR, each independently encoded rate is considered a separate layer. The notation above holds so long as nothing as been downloaded for a chunk ($m[t,n] = 0$), since for those chunks ΔR and ΔD are still correct.

A. Obtaining the Rate Constraints

First, we derive the rate constraints for the optimization. Although the goal of the optimization is to find an allocation in bits to meet the rate constraints, it is actually simpler to come up with the rate constraints in units of time rather than bits and then translate those constraints to bits. At time t, suppose we spend time $T[n]$ downloading information for chunk n (for example, suppose we allocate 500Kbits to a chunk over a 1Mbps link, then $T[n] = 0.5$ sec). To prevent glitching during playback and for the information downloaded for chunk n to be useful, the only constraint we need to satisfy is

$$\sum_{i=0}^{n} T[i] \leq p[n] - t, \tag{2}$$

for $n = 0, 1, \ldots, N-1$. That is $T[0] \leq p[0] - t$, $T[0] + T[1] \leq p[1] - t$, and so on. Thus the total time we can spend downloading all chunks up to some chunk is its playback time minus the current time. This constraint needs to be satisfied for those chunks which have not yet been played (chunks where $p[n] > t$). At the start, since $t = p[0] - \kappa$, alternatively we can write $T[0] \leq \kappa$, $T[0] + T[1] \leq \kappa + d[0]$, and so on.

Contrary to what is commonly done, we don't place any additional constraints on decoder buffer overflow (equivalently encoder buffer underflow). Such constraints only make the quality suboptimal. There is no advantage to limiting the decoder buffer size as there is plenty of disk space available - download and play or progressive download already use a very large decoder buffer (the file size itself). Even for streaming of a live event, we can only optimize up to the last chunk which has been encoded, so encoder underflow is also not an issue.

In addition to the simple constraint in equation (2), we would like the buffer to be of a certain size or follow some schedule so that eventually we have sufficient buffer to handle network variability in bandwidth. As we download information for the chunks, the buffer in Eqn. (1) evolves as

$$B\left[t+\sum_{i=0}^{n} T[i]\right]=\begin{cases} p[Z[t]]-(t+\sum_{i=0}^{n} T[i]) \text{ if } n<Z[t] \\ p[n+1]-(t+\sum_{i=0}^{n} T[i]) \text{ if } n\geq Z[t] \end{cases} \quad (3)$$

that is it decreases so long as we are downloading additional enhancement layers for chunks which have already been downloaded and then increases or decreases depending on the time spent downloading new chunks with respect to their duration. An example of this evolution is shown in Fig. 3.

To prevent sudden jumps in media quality, we can ramp up or down the buffer to achieve the desired buffer by some time into the future. During the ramping of the buffer, we can define $\beta[n]$ as the buffer we would like to have by the time all the information for chunks up to and including n has been downloaded, that is we want $\beta[n]$ to be larger than or equal to the quantity in equation (3). From this,

$$\sum_{i=0}^{n} T[i] \leq \begin{cases} p[Z[t]]-t-\beta[n] & \text{if } n<Z[t] \\ p[n+1]-t-\beta[n] & \text{if } n\geq Z[t] \end{cases} . \quad (4)$$

To obtain $\beta[n]$, we can simply use a linear ramp between the current buffer size $B[t]$ and the desired buffer size B_D, as

$$\beta[n]=\begin{cases} B_D, & \text{if } p[n+1]\geq t+B_T \\ B[t]+\frac{(p[n+1]-t)}{B_T}(B_D-B[t]), & \text{else} \end{cases}, \quad (5)$$

where B_T is the time we can take to reach the desired buffer.

Now, we can convert the constraints on $T[n]$ obtained above to bit constraints. Let $r[i]$ be the number of bits that are allocated to chunk i. Then, if the current estimate of the bandwidth is W bits per unit of time, combining the results from (2) with (4) gives the final multiple rate constraints $C[n]$ (in units of bits) as

$$\sum_{i=0}^{n} r[i] \leq \max(0, C[n])$$

$$C[n]=\begin{cases} W\min(p[Z[t]]-\beta[n], p[n])-t, \\ \text{if } n<Z[t] \\ W\min(p[n+1]-\beta[n], p[n])-t, \\ \text{if } n\geq Z[t] \end{cases} \quad (6)$$

for $n=0,1,\dots,N-1$. Note that the rate constraint cannot be negative and from now on, assume that $C[n]$ refers to $\max(0, C[n])$. The first argument in the $\min$ is to meet the desired buffer constraint (4) and the second is to meet the playback deadline constraint (2).

In practice, since we don't know the bandwidth of the link, we estimate it by updating as each download completes as $W_k=W_{k-1}\alpha+\frac{P_k}{U_k}(1-\alpha)$, where W_k is the estimate after the kth download, P_k is the size of the kth download, U_k is the time taken for the kth download, and α is a decay factor. α can also be a function of the relative change, for example, if P_k/U_k is very different from the previous estimate, we can reduce α so that the new information has more weight.

B. Solving the Optimization Problem

The goal of the optimization is to minimize the total distortion $\sum_{j=0}^{N-1} D[j]$ subject to the multiple rate constraints in (6). Since chunk n is operating at layer $m[n]$ (the number of layers it has already downloaded), the objective becomes to find the next operating point for it, $m'[n]$. For simplicity, we drop the subscript t from $m[t,n]$. We find $m'[n]$ to minimize

$$\sum_{j=0}^{N-1} D[j, m'[j]] \text{ subject to}$$
$$R_T[n]=\sum_{i=0}^{n} r[i] \leq C[n], \quad (7)$$

for $n=0,1,\dots,N-1$, where $r[i]=R[i,m'[i]]-R[i,m[i]]$ is the additional rate used by chunk i to go from operating at layer $m[i]$ to layer $m'[i]$, and $R_T[n]$ is the cumulative additional rate used by all chunks up to chunk n.

The solution to the multiple rate constraints problem may seem difficult but can be solved with reasonable complexity. We rediscovered an algorithm which is similar to [6]. The way it works is to first solve (7) subject to the single rate constraint for $n=N-1$, that is minimize the total distortion subject to the final rate constraint. Then the constraint in (7) is satisfied for $n=N-1$. However, the constraint may be violated for some other $n<N-1$. We find the first n for which the constraint is violated, $I_0=\min\{n: R_T[n]>C[n]\}$. If none exists, then we are done. However, if there is a violation for some I_0, then we solve the following two problems. First we solve the single rate constraint problem to minimize

$$\sum_{j=0}^{I_0} D[j, m'[j]] \text{ subject to}$$
$$R_T[I_0]=\sum_{i=0}^{I_0} r[i] \leq C[I_0] \text{ and then minimize,} \quad (8)$$
$$\sum_{j=I_0+1}^{N-1} D[j, m'[j]] \text{ subject to}$$
$$R_T[n]-R_T[I_0]=\sum_{i=I_0+1}^{n} r[i] \leq C[n]-R_T[I_0], \quad (9)$$

for $n=I_0+1,\dots,N-1$. After solving the first problem (8), we are guaranteed to not have violations of the constraint in (7) for $n\leq I_0$. This is because if $R_T[I_0]>C[I_0]$ before, then solving the first of the two problems does not increase $r[i]$ (and thus $R_T[i]$) for $i\leq I_0$. Since we had no violations of the constraint for $i<I_0$, we will continue to have none. The second problem is the same as the original (7), but with modified multiple rate constraints. If there are additional violations in the solution to the second problem, we repeat the

process. Since the number of rate constraints in the second problem decreases by at least one, we will reach a solution.

The only issue remaining is to come up with a method for solving the single rate constraint problem, i.e. solve (7) subject to the rate constraint for some n. This is a classical problem that has been well studied. If there is a very fine grained scalable coding where we can obtain any arbitrary bitrate, then the solution is found by minimizing the Lagrangian $J = \sum_i (D[i] + \lambda R[i])$ and then finding the λ which meets the rate constraint [9]. This results in the constant slope solution, that is all the chunks operate at the same $\lambda = -dD/dR$ point.

For coarser scalable and MBR codings, the best solution is difficult to find and can be done by an exhaustive search (high complexity), a search around the approximately constant slope point, or other reduced complexity suboptimal dynamic programming approaches. Here we use a simple algorithm which is not necessarily optimal, but works reasonable for our case. The idea is to allocate bits greedily by picking those chunks which will have maximum reduction in distortion for the increase in rate. Let Q be the current number of bits that have to be allocated starting with $Q = C[n]$. First, find the chunk i which has the maximum distortion-rate slope at the current operating point over a set of chunks. That is find, $i = \arg\max_{k \in \mathcal{L}} S[k, m[k] + 1]$, where $S = -\Delta D/\Delta R$, and $m[k]$ is the current number of layers downloaded for chunk k. The search is done over $\mathcal{L} = \{j : \Delta R[j, m[j] + 1] \leq Q\}$, that is the set of chunks which can go to the next layer using no more than the current available Q bits. Then, set $Q \leftarrow Q - \Delta R[i, m[i] + 1]$ (reduce the number of bits), and $m[i] \leftarrow m[i]+1$. This process is repeated until no chunk exists which can go to the next layer using no more than Q bits, $Q = 0$, or until all chunks reach their maximum layer. Using a sorted list of slopes can allow this allocation to be solved with a single pass through the list. Since the optimization is done at the chunk level, the complexity is insignificant.

Suboptimality in the solution occurs when the i with the maximum slope has ΔR larger than the remaining Q bits. In these cases, sometimes modifying previous allocations to free up rate to accommodate this may be more optimal. This may also result in situations where there is residual rate from the constraint. Small searches around the obtained solution can be used to improve the allocation, but may result in significant complexity. However, any residual rate is not wasted and can be used for download of future chunks.

Since the bandwidth estimate is updated after each download, it changes the rate constraints, and thus we reevaluate the rate allocation. Therefore, it is not useful to perform the rate allocation for all future chunks at once. If only q downloads are to be started, then only the next q allocations need to be found, for which we can show we need to solve the single rate constraint problem at most $2q$ times.

The same solution can be used for MBR with the exception that updating previously downloaded chunks with a new bitrate encoding is not correct because the R-D curve is no longer convex (this is rare anyways). However, for determining the initial download rate, the solution is still correct.

C. Further Reducing Startup Latency

From equation (2), note that the minimum startup delay for playback is given by the time needed to download the first chunk at the lowest bitrate. Here we present two potential ways to reduce this further. Since both of these result in suboptimal encodings for the chunk (when compared to VBR coding of the entire chunk), we can have these encodings available for all chunks, although to improve compression efficiency, they should only be used for the first few chunks.

First, we can subdivide a chunk into smaller duration subchunks as shown in Fig. 1 (e.g. each 2 second chunk can be divided into two 1 second subchunks). If the subchunks are available at multiple quality levels, they can be factored into the optimization without any modification to the constraints.

Alternatively, we can have additional CBR encodings available as shown in Fig. 1 which can relax the requirement that the entire chunk be downloaded prior to starting playback of the chunk as needed for VBR encodings. Suppose the CBR encoding is done so that for chunk n, the information that we are going to download only needs a buffer of $\tau[n]$ (in units of time) prior to starting playback at the current network bandwidth - that is it is a CBR encoding with a small buffer. If the CBR encoding is one where each frame takes exactly the same number of bits, then $\tau[n]$ can be as small as one frame. Now, we can potentially download $d[n] - \tau[n]$ of the chunk after the playback time $p[n]$. If we need time $T[n]$ to download information for chunk n, then we need time $T[n](d[n] - \tau[n])/d[n]$ to download this portion of the information for the chunk. If the bitrate being downloaded is less than the network bandwidth ($T[n] <= D[n]$), then we can modify the constraints in (2) as $\sum_{i=0}^{n} T[i] \leq p[n] - t + T[n] \left(1 - \tau[n]/d[n]\right)$, which gives, $\sum_{i=0}^{n-1} T[i] + T[n] \frac{\tau[n]}{d[n]} \leq p[n] - t$.

III. Evaluation on Video Content

First we divide the video into two second chunks and then encode each chunk using the JSVM 9.12 Scalable Video Codec (SVC) using medium grain scalability. From this encoding, we extract streams from 500Kbps to 2Mbps in increments of 100Kbps which gives the streams which are available for download for the chunks. For each bitrate, we decode and compute the average mean square error for the chunk which is used as the distortion metric. The video is 720x480 at 30 frames/sec. Two test clips are used, one a high motion sports clip, and another a low motion television show, both approximately four minutes in length giving about 120 chunks. We do not use the standard test sequences as their length is too short to obtain a reasonable number of chunks. Results are presented for the luminance (Y) component of the video only.

We test the algorithm performance by simulating transmission over a channel with fixed and variable bandwidth and present results comparing our R-D optimized algorithm with a pure rate based adaptation strategy as is commonly used. In Fig. 4, we show results for the sports clip when operating over a mostly fixed 750Kbps connection, with $\pm 10\%$ bandwidth variability, with an initial startup delay of one second, and the desired buffer being six seconds (three chunks). In Fig. 4(a),

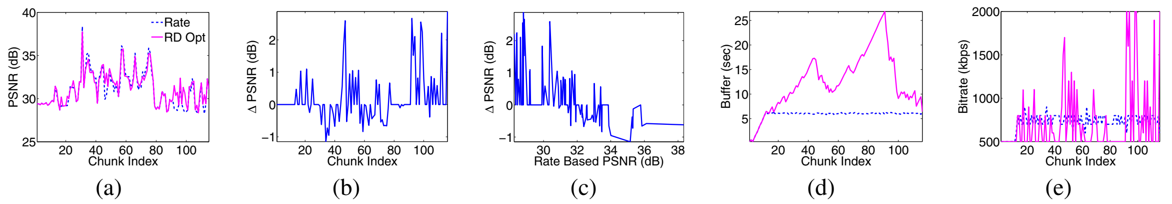

Fig. 4. Results for sports clip at fairly constant network bandwidth (750Kbps±10%) – in (a), (d), (e), dash line is rate based, solid is R-D optimized.

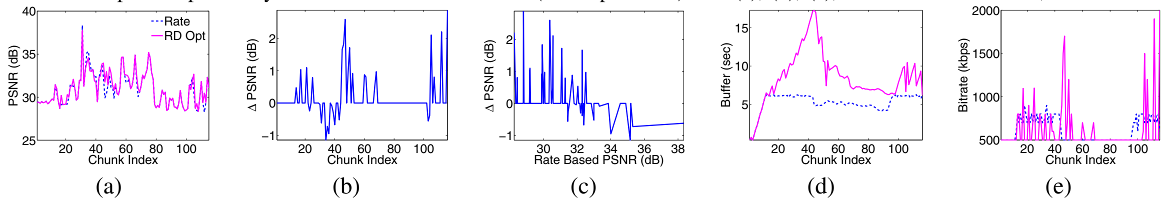

Fig. 5. Results for sports clip at variable network bandwidth - switching from 750Kbps to 500Kbps back to 750Kbps.

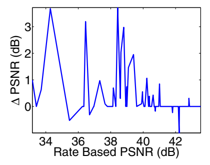

Fig. 6. PSNR imp. for drama clip.

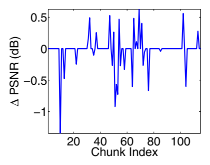

Fig. 7. PSNR diff. reducing delay.

we show the average PSNR for each chunk for both strategies and in Fig. 4(b) shows the gains when using R-D optimization. The overall gain across all chunks is 0.3dB PSNR. Although the overall gain is small, the maximum gain across all chunks is quite large, close to 3dB. To show which chunks achieve the most gain, in Fig. 4(c), we plot the PSNR gain vs. the PSNR achieved from the rate based adaptation strategy. As expected, the large gains are for those chunks which otherwise would have a low PSNR using the rate based adaptation strategy. For those chunks which have a larger PSNR, there is a slight loss. This is expected as the R-D optimization will essentially take bitrate from regions which are easier to code, lowering their PSNR, and give them to regions which are harder, improving their PSNR. Fig. 4(d), shows the client buffer fullness. By downloading chunks at mostly a constant rate, the pure rate based adaptation strategy always has a buffer close to the desired buffer size after the initial ramp up. The R-D optimized strategy on the other hand has a variable buffer which is sometimes much larger than the desired, which allows a larger bitrate to be used for some chunks in the future. Fig. 4(e) shows the average bitrate of the chunks being downloaded. Again, the rate based strategy always downloads chunks close to the network bandwidth whereas the R-D optimized one downloads at variable bitrates and uses a higher bitrate on the more difficult chunks. The overall effect is a smoother video, with the quality being more constant rather than the bitrate.

In Fig. 5, we repeat the experiment but with variable bandwidth. For the first 50 chunks the bitrate is 750Kbps with $\pm 10\%$ variation, drops to 500Kbps for the next 50 chunks, and then returns to 750Kbps. We see similar results with the maximum PSNR improvement being about 3dB.

In Fig. 6, we show results for the television drama clip using variable bandwidth, showing only the PSNR improvement vs. the rate-based adaptation PSNR. Again, we see that some of the lower quality chunks have a large improvement, close to 4dB, with the higher PSNR chunks having a slight reduction. The overall gain across all chunks is about 0.3dB.

In Fig. 7, we show how PSNR is affected by using smaller subchunks for the initial two chunks, each chunk being subdivided into two subchunks as Fig. 1. This reduces the initial startup latency from 1.0 second to 0.5 seconds. We see that there is only one initial chunk which has a reduction in PSNR of 1dB, all other chunks have PSNR differences less than 0.5dB. By only using the split chunks at the start, we reduce the startup delay, but only affect the quality at the beginning.

IV. Conclusion

We presented a solution to improve quality when streaming media over unreliable networks such as the Internet. Although chunked adaptive streaming technologies are becoming popular, existing bitrate adaptation techniques do not take advantage of content variability over the chunks. By using flexible encodings and an R-D optimized bitrate adaptation technique, we can significantly improve quality on difficult portions of the content and reduce quality variations.

References

[1] G. Conklin, G. Greenbaum, K. Lillevold, A. Lippman, and Y. Reznik, “Video coding for streaming media delivery on the internet,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 269–281, Mar. 2001.

[2] C. Huang, P. Chou, and A. Klemets, “Optimal coding rate control of scalable and multi bit rate streaming media,” *Microsoft Research Technical Report MSR-TR-2005-47*, Apr. 2005.

[3] D. Ye, Q. Wu, and Z. Zhang, “Receiver-buffer-driven approach to adaptive internet video streaming,” *Electronics Letters*, vol. 38, no. 22, pp. 1405–1406, Oct. 2002.

[4] Move Networks - Move Adaptive Stream. [Online]. Available: http://www.movenetworks.com/move-media-services/move-adaptive-streaming

[5] Adaptive Streaming. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd159743.aspx

[6] A. Ortega, “Optimal bit allocation under multiple rate constraints,” in *Proc. Data Compression Conference*. Snowbird, UT: IEEE Computer Society, Apr. 1996, pp. 349–358.

[7] D. Lin and J.-J. Chen, “Efficient optimal rate-distortion coding of video sequences under multiple rate constraints,” in *IEEE Trans. Image Processing*. IEEE, Oct. 1997, pp. 29–32.

[8] L. Zhao and C.-C. J. Kuo, “Buffer-constrained R-D optimized rate control for video coding,” in *Proc. Int’l Conf. Multimedia and Expo*. IEEE, Jul. 2003, pp. 377–380.

[9] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: John Wiley and Sons, 1991.