Multiple Distortion Measures for Packetized Scalable Media

Carri W. Chan, *Student Member, IEEE,* Susie Wee, *Member, IEEE,* and John Apostolopoulos, *Fellow, IEEE*

***Abstract*—As the diversity in end-user devices and networks grows, it becomes important to be able to efficiently and adaptively serve media content to different types of users. A key question surrounding adaptive media is how to do Rate-Distortion optimized scheduling. Typically, distortion is measured with a single distortion measure, such as the Mean-Squared Error compared to the original high resolution image or video sequence. Due to the growing diversity of users with varying capabilities such as different display sizes and resolutions, we introduce *Multiple Distortion Measures* to account for a diverse range of users and target devices. Multiple Distortion Measures (MDM) gives a clear framework with which to evaluate the performance of media systems which serve a variety of users. Scalable coders, such as JPEG2000 and H.264/MPEG-4 SVC, allow for adaptation to be performed with relatively low computational cost. We show that accounting for MDM can significantly improve system performance; furthermore, by combining this with scalable coding, this can be done efficiently. Given these Multiple Distortion Measures, we propose an algorithm to generate *embedded* schedules, which enables low-complexity, adaptive streaming of scalable media packets to minimize distortion across multiple users. We show that using MDM achieves up to 4dB gains for spatial scalability applied to images and 12dB gains for temporal scalability applied to video.**

***Index Terms*—Multiple distortion measures, scalable streaming, embedded packet schedules, rate-distortion optimization, JPEG2000, H.264/MPEG-4 SVC**

I. Introduction

Multimedia delivery systems encode multimedia content into packets that are sent over a network to one or more receivers, and receivers receive some or all of these packets depending on network congestion and packet loss. A critical part of a multimedia delivery system is the scheduling algorithm that the sender uses to determine which multimedia packets to prioritize and send over the network. Much prior work has been done to find scheduling algorithms that optimize the rate-distortion performance of the delivery system. In much of this work, each packet has an associated incremental rate (size) and incremental distortion value (e.g., mean-squared error) that it contributes to the reconstruction of the multimedia content. The incremental rate and incremental distortion value of each packet can be used to determine the relative importance of the packets, and scheduling decisions can be made according to this information.

Manuscript received September 7, 2007; revised June 2, 2008.

C. W. Chan is with the Department of Electrical Engineering, Stanford University, Stanford, CA, 94305 USA e-mail: cwchan@stanford.edu.

S. Wee is with Hewlett-Packard, Cupertino, CA 95014 USA e-mail: susie.wee@hp.com.

J. Apostolopoulos is with Hewlett-Packard Laboratories, Palo Alto, CA 94304 USA e-mail: john_apostolopoulos@hp.com.

Digital Object Identifier ??

Traditionally, the incremental distortion value of each packet is computed in relation to the original multimedia content, and each multimedia packet has a single distortion value associated with it. Traditional scheduling algorithms use a single distortion measure, such as mean-squared error in relation to the original image. However, multimedia delivery systems are increasingly serving many receivers with a diverse range of characteristics. For example, display devices for images and video range from cellphones to PDAs to PCs, each with capabilities for different sizes, resolutions, and framerates. Scalable media, such as JPEG2000 [1] and H.264/MPEG-4 SVC [2], has the capability to adapt to different user types and helps address this issue. However, scalable media tends to give a coarse granularity for adaptation and, as we will later show in Section III, it does not necessarily provide optimal performance. In order to account for each user's capabilities, it may be appropriate to customize a different distortion measure for each device, e.g., for a low-resolution display device the mean-squared error should be computed in relation to a low-resolution version of the original image.

In this paper, we propose using *Multiple Distortion Measures* (MDM) to explicitly account for the diversity of receivers in today's multimedia delivery systems. Within the MDM framework, each multimedia packet has multiple distortion values associated with it, one for each chosen distortion measure. Scheduling algorithms can then be developed using these multiple distortion measures, specifically, using the incremental rate and the multiple incremental distortion values of each packet. To our knowledge, this class of algorithms has not yet been explored.

Two questions that arise are: 1) What is the difference in the optimal schedule for different distortion measures, and 2) What is the benefit of using multiple distortion measures in packet scheduling algorithms? While one might intuitively expect some difference in the distortion values of multimedia packets, one might expect the relative importance of packets to be quite similar. A surprising result we found was that the difference in the relative importance of packets can be quite large for different distortion measures. We show these results in Section III. Furthermore, by using multiple distortion measures we were able to develop scheduling algorithms that achieve significantly improved performance over those that only consider a single distortion measure. We show results for this in the context of images in Sections IV and in the context of video in Section V.

A. Related Work

With the growing diversity of mobile devices, a significant body of work has been done to develop effective methods to serve media content to multiple user types.

When the capabilities of all clients are known a priori, one can encode a scalable stream to optimally serve them all. A large body of work has looked at how to adapt frame rate and modify encoding for different users (see [3]–[8] and related work). In all of this work, the benchmark for performance (if one exists) is the original sequence–an approach which may neglect different user types. In [9], the authors look at how to encode for multiple spatial resolutions. In this case, they use CIF and QCIF benchmark video sequences to evaluate performance of the scalability. There has been a substantial amount of work focused on modifying frame rate, image quality, and spatial resolution at the encoder in order to serve different types of users. It is possible to encode the media sequence to adhere to the specific needs of each user type, if they are known. However, when the bitrate and user type of a particular client is not known at the time of encoding, how can a service provider adapt the media content to satisfy these constraints in a visually pleasing manner?

If adaptation is necessary after encoding, it is possible to decode and re-encode the media sequence in order to adhere to the rate constraint and viewing needs of each particular user. Unfortunately, this can be extremely expensive in terms of computation time and power. Rather, transcoding can be done to modify a (non-scalable) coded sequence into a different coded sequence with different properties, such as bitrate, frame rate, spatial resolution, etc. An overview of transcoding can be found in [10]–[12]. In [13], [14], the authors look at how to transcode pre-encoded video into video with lower spatio-temporal resolution. While these works adapt to the various display capabilities of different users, by upsampling and interpolating, they also focus on the original high, rather than low, resolution display as the benchmark for performance. In this paper, we will show that modifying the benchmark image/video leads to significant gains.

With the growth of scalable codecs such as JPEG2000 and H.264/MPEG-4 SVC, transcoding operations can be simplified to truncations of bitstreams, and discard or truncation operations of packets. This makes it possible to encode the media once and adapt it to user demands without expensive re-encodings or transcoding operations. We focus on this problem of developing scheduling algorithms to jointly optimize transmission of packetized scalable media, where the bitstream can be altered by discarding packets. We assume that we are given an encoded bitstream and we wish to transmit it to multiple types of users. We look at how to evaluate performance as well as how to generate embedded packet schedules given the new framework. Embedded schedules are schedules which build upon themselves. They are useful because they reduce transcoding operations to simple truncations of the codestream, and also allow for meaningful transcoding even when the codestreams are encrypted [15]. Optimized packet scheduling is an important problem and has garnered quite a lot of attention. Some of the early work includes [16]–[19] and [20], [21] for embedded scheduling. We refer the reader to the preceding references and the references therein for more background on packet scheduling.

Given this prior work, and using the conventional approach of a single distortion measure, one possible approach is to generate many schedules–one for each user type. However, this could be costly for a number of reasons. First is the memory required to store all the different schedules, rather than just one which can serve many. Also, suppose a media stream were transmitted to a relay node before adapting it to specific users needs–it is unrealistic to require complex transcoding operations on a per-user basis at the relay. Finally, if the media stream is encrypted, using an embedded schedule jointly optimized for multiple users, given MDMs, allows for adaptation without requiring access to the unencrypted stream. We will explore the scheduling dilemmas–specifically in the case of embedded scheduling–that arise due to the conflicts in prioritization of packets for each user.

Most closely related to our work, [22] examines temporal and spatial adaptation of scalable video and how to evaluate performance of scalable media. Thus far, it has been difficult to evaluate the relationship between scalable operations and viewer utility of the resulting stream. In their work, the authors propose using a user/classification-based performance metric for quality assessment. Through subjective tests and a machine-learning, predictive framework, they are able to evaluate performance of scalable video systems. In our work, an extension to [23], we propose a general framework in which to evaluate the performance of scalable systems and how to optimize embedded schedules of scalable media packets. Our goal is to minimize the associated distortion for each user. In the remainder of this paper, we discuss how to customize new distortion measures which accurately capture the specific needs of each user and how to schedule packets given these measures. Our contribution is the introduction of a framework which we call Multiple Distortion Measures (MDM). The main distinction between our work and [22] is the generality of our work–we can incorporate their utility functions determined through subjective studies–as well as our study of how to do multi-objective scheduling given these multiple user types. We introduce a clear framework which is robust and independent of highly variant user opinions and allows the use of the standard measure of mean-squared error (MSE) distortion. Customizing a distortion measure for each user type leads to substantial gains, due to the surprising variance in media packet importance depending on the user and device type.

The rest of the paper is organized as follows. In Section II, we present the general framework in which Multiple Distortion Measures is formally defined. In Section III, we apply the MDM framework to a specific instance of images where distortion measures are defined by spatial resolution. We use this instance to gain insight about Multiple Distortion Measures. In Section IV, we look at the scheduling problem of generating embedded schedules involving media packets with multiple distortion measures defined by resolution. We develop a scheduling algorithm and use the framework from Section II-B to evaluate performance via empirical experimentation. In Section V, we discuss the generality of our framework

Application	Types of MDM
Image	Resolution PSNR fidelity Color fidelity
Video	Resolution PSNR fidelity Color fidelity Frame Rate
Audio	Bandwidth # of channels (mono/stereo)
Graphics	Shape Texture

Fig. 1. Summary of some potential applications of Multiple Distortion Measures.

with an extension to temporal scalability of video. Finally, we conclude in Section VI.

II. Multiple Distortion Measure Framework

This section formally introduces the Multiple Distortion Measures framework. Packetized scalable media allows for adaption beyond the original high resolution image or original high resolution, high frame rate video by simply discarding select packets of the encoded bitstream. Typically, each user is most concerned with metrics that impact his own performance. For instance, a low resolution viewer cares about distortion and PSNR compared to a low resolution image, rather than the original high resolution image–a resolution he cannot view. However, if a single metric is used based on the high resolution image, then the needs of the low resolution viewer could be ignored. For this reason, we introduce *Multiple Distortion Measures* to account for and measure performance relative to multiple user types.

With the growing diversity in multimedia devices, it is generally the case that users will view content on different types of displays. Therefore, we generate multiple benchmark images/videos which incorporate the various display capabilities of each user type and which are used to measure the distortion of a reconstructed image or video sequence. For instance, a benchmark image/video could be a downsampled, low resolution version of the original; a grayscale version of the original, 3 color component image/video; a temporally downsampled version of the original video sequence; or a highlighted Region-of-Interest (ROI). Table 1 summarizes a few potential application areas and capabilities which Multiple Distortion Measures could help account for multiple user types.

A. Defining Multiple Distortion Measures

A key aspect of MDMs is calculating the different distortion measures. This involves selecting a distortion metric, such as mean-squared error or mean-absolute difference, and an appropriate reference. The reference can be an appropriate transformation of the original content. We define the transformation and distortion metric in this section.

We define by $\mathcal{T}_u(X)$ a *transformation* operator of media content, X, for user type u. A transformation converts media content X into a modified, benchmark version which user type u will view and consume the content. For example, this transformation could be spatial downsampling to convert our original benchmark image, X, into a low resolution benchmark image, $\mathcal{T}_u(X)$, if user type u wishes to view the image on a low resolution display. The transformation could also be a temporal downsampling or framerate conversion operation, such as frame dropping, to reduce the frame rate for video. Therefore, $\mathcal{T}_u(X)$ is the reference media against which performance evaluation is measured for user i. Define $\mathcal{T}_I$ as the identity transformation such that $\mathcal{T}_I(X) = X$. There will be multiple transformation operators–one corresponding to each user type.

These multiple benchmark images/videos (one for each transformation) are now used to calculate distortion values of the reconstructed media–hence, Multiple Distortion Measures (MDM). Let's define $D_u(\hat{X})$ as the distortion of reconstructed media $\hat{X}$ compared to the benchmark media $\mathcal{T}_u(X)$. Note that this is a function of X and $\hat{X}$ as well as the transform $\mathcal{T}_u$, $D_u(\hat{X}) = f(\hat{X}, \mathcal{T}_u(X))$. When the received media, $\hat{Y}$, is displayed and reconstructed differently, it may be difficult to make a comparison to the original content, X, since the reconstructed media and benchmark media have different resolutions or frame rates. This is sometimes bypassed by up-sampling a low resolution/low frame rate image/video. Instead of calculating distortion of $\hat{Y}$ compared to the original benchmark X for all users, we propose to calculate distortion compared to a transformed benchmark media, $\mathcal{T}_u(X)$, specialized for each user type. This provides a more applicable performance evaluation across multiple users. In the scenario of Fig. 2 with a low and high resolution user, there would be 2 distortion measures defined by 2 benchmark images: one high resolution benchmark, $\mathcal{T}_I(X) = X$, and one low resolution benchmark, $\mathcal{T}_L(X) = Y$.

These distortion measures may be applied at different levels of granularity. For example, distortion values may be computed for packets of packetized multimedia content such that multiple distortion values are calculated for a single packet. Furthermore, the distortion values may be calculated with respect to different benchmarks, e.g., for image or video, a packet's value can be calculated in relation to the low resolution and high resolution reconstructions.

B. Embedded Scheduling for Packetized Media with Multiple Distortion Measures

A schedule is an ordering of packets that indicates how they should be sent over a network. Embedded schedules have the characteristic that all packets included at rate R_1 are also included at rate $R_2 > R_1$. That is, embedded schedules incrementally add packets for increased rates.

Given a single distortion measure, an embedded schedule can be determined in a rate-distortion optimized manner [20], [21]. The extension of this algorithm to packetized scalable media with Multiple Distortion Measures is not obvious. Our scheduling goal is to generate *embedded* schedules in the context of Multiple Distortion Measures.

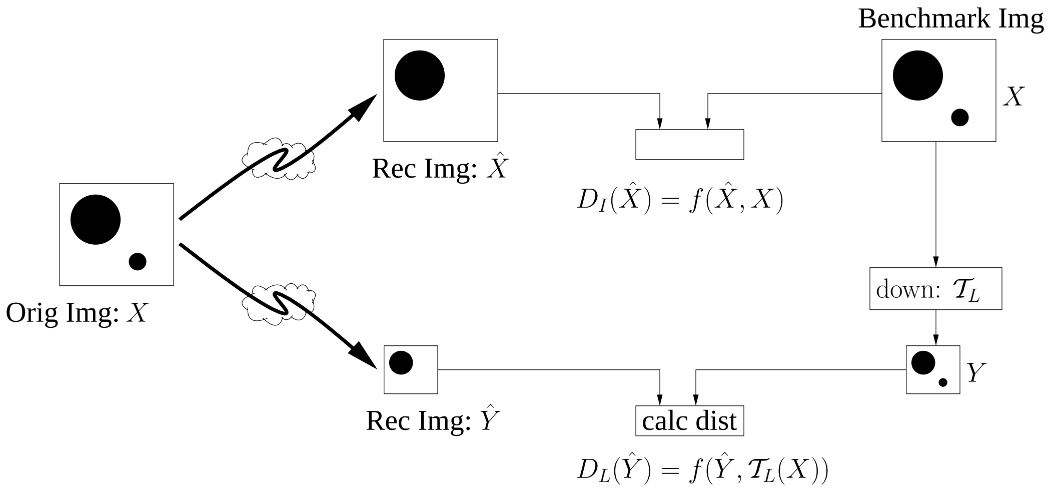


Fig. 2. Diagram of Multiple Distortion Measures. Multiple benchmark images (or videos) are generated: the original, $\mathcal{T}_I(X) = X$, and a *transformed* one, $\mathcal{T}_L(X)$. Distortion of reconstructed images is compared to these multiple benchmarks which are able to accurately capture the display capabilities of the particular user in question. In this case, the low resolution benchmark image $Y = \mathcal{T}_L(X)$ is a downsampled version of the original benchmark image.

Embedded schedules allow for simple transcoding operations with a simple truncation of the bitstream. With a single distortion measure, the optimal embedded schedule can be found via the fused-greedy algorithm, described in Section II-C. However, Multiple Distortion Measures introduces different quality measures for different devices and users. The media packet importance varies depending on the distortion measure used. This creates a conflict when simultaneously serving multiple users with different distortion measures. The fused-greedy algorithm is not applicable for multiple distortion measures, thus alternative scheduling algorithms for multiple distortion measures are presented in Section IV. This section will discuss the framework for embedded scheduling with multiple distortion measures.

The goal is to build the MDM-aware scheduler in the system depicted in Fig. 3. We want to adapt a precoded scalable media stream to serve multiple user types at various rate constraints. Let $\mathcal{U}$ be the set of user types. In Fig. 3, we depict low and high resolution user types–each with N_u users. Each user type, u, will consume the media at some rate $0 \leq R_u \leq R_{\text{max}}$. In this case, $\mathcal{T}_u(\cdot)$ is the transformation benchmark for user type $u \in \mathcal{U}$. Let p be the probability distribution function for media consumption, so that $p(u, R)$ is the probability that user type u views the image/video at rate R. Our goal is to design the MDM scheduler to schedule the packets of the scalable media in order to minimize the distortion over the diverse set of clients.

A natural performance metric to optimize over is expected weighted distortion, or expected distortion where all the weights are equal to 1. These weights are useful to allow varying prioritization of different user types. For instance, if one user type is willing to pay more for better viewing quality, it may be useful to weight his distortion contribution more heavily in order to ensure it is small. A schedule defines, for each rate, a subset of packets of the encoded bitstream which adhere to the rate constraint. Let S denote a schedule and $S(R)$ is the reconstructed content of the schedule with rate constraint, R. Then, given weights $w_{u,R}$, the expected weighted distortion for schedule S is:

$$E_w[D|S] = \sum_{u \in \mathcal{U}} \sum_{R=0}^{R_{\text{max}}} p(u, R) w_{u,R} D_u\big(S(R)\big) \tag{1}$$

Given this performance metric, we have a framework in which to compare schedules. If $E_w[D|S_1] < E_w[D|S_2]$, then we can say schedule S_1 is better than schedule S_2.

We focus on the scheduler part in Fig. 3, which we examine more closely in Fig. 4. The scheduler is given the distribution of user types and rate constraints, $p(u, R)$, as well as the transforms, $\mathcal{T}_u$, which define the MDM to make scheduling decisions. Conventional systems incorporate only a single distortion measure, $\mathbf{d}_1$, to make scheduling decisions. They assume that there is only one user type, so that $p(u, R) = p(R)$. By introducing multiple benchmarks defined by transforms, $\mathcal{T}_2, \mathcal{T}_3, \ldots, \mathcal{T}_u$, we generate multiple lists of distortions, $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_u$, which define the importance of each packet to each user type. The list $\mathbf{d}_u$, generated by transform $\mathcal{T}_u$, consists of values, $d_{u,i}$, which is the amount of distortion incurred by the loss of packet i when distortion is measured against the benchmark image defined by $\mathcal{T}_u$. Now, instead of each packet having a single distortion value, d_i, each packet has multiple distortion values $(d_{1,i}, d_{2,i}, \ldots, d_{u,i})$ corresponding to each user type and associated transformation operator, $\mathcal{T}_u$. Note that these distortion lists can be generated during or after encoding. Our goal is to design the scheduler which incorporates the MDM information provided by the analysis to generate an ordered set of packets from the original set of packets provided by a scalable encoder.

Define S_L as the schedule optimized for low resolution viewing. Let R_{max} be the highest possible viewing rate; for instance, the size of the original coded image. In this case, we assume only 1 type of user, $\mathcal{U} = \{L\}$, and a uniform distribution of rates, $p(L, R) = \frac{1}{R_{\text{max}}}$, i.e., the probability of viewing the image at rate $r \in [1, R_{max}]$ is uniform and all

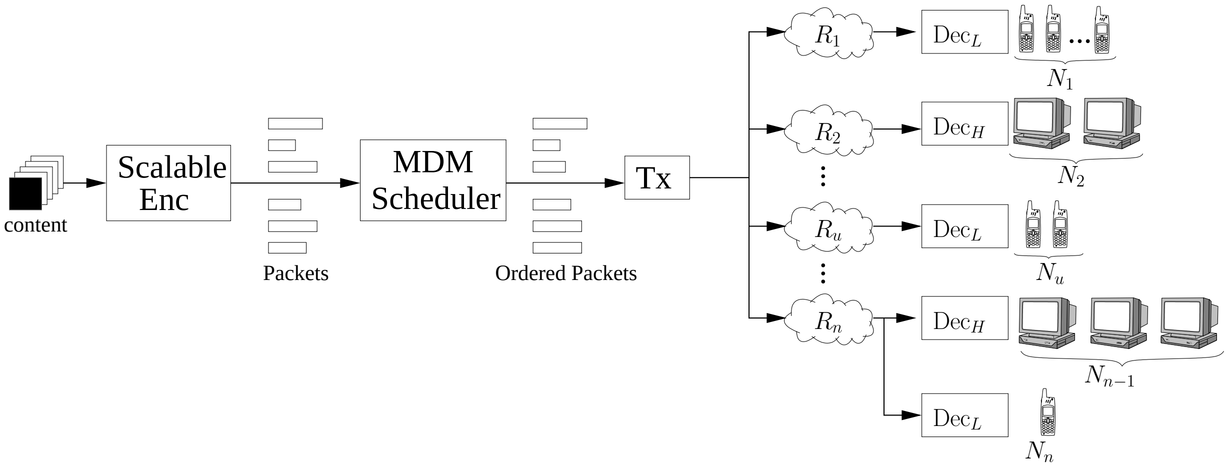


Fig. 3. System diagram for scheduling problem. We want to design the Multiple Distortion Measure (MDM) scheduler to order scalable media packets to serve multiple types of users over multiple rate constraints. The rate constraints and user types can be known deterministically or probabilistically.

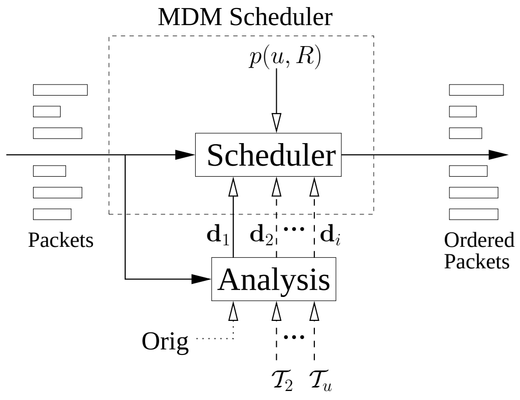


Fig. 4. Our scheduling problem focuses on the MDM scheduler. The scheduler is given multiple transforms to define benchmark images/videos for each user type as well as the channel and user type distributions. Given this information, it orders the packets into a MDM-aware schedule. Note that the original sequence may not be available, so it may be estimated as the decoded sequence at the highest possible quality.

users have low resolution displays. If we let $\mathcal{S}$ be the set of all possible embedded schedules, then S_L can be defined as:

$$\begin{aligned} S_L &= \arg\min_{S\in\mathcal{S}} E_w[D|S] \\ &= \arg\min_{S\in\mathcal{S}} \sum_{R=0}^{R_{\max}} \frac{1}{R_{\max}} D_L\big(S(R)\big) \end{aligned} \tag{2}$$

This optimal schedule can be determined using the fused-greedy algorithm of [20]. Analogously, we can define S_H, the optimal schedule for high resolution viewing.

C. Fused-Greedy: Embedded Scheduling for a Single Distortion Measure

We now briefly review the algorithm to generate embedded schedules developed in [20]. This algorithm generates the optimal embedded schedule for a *single* distortion measure, assuming a uniform distribution of weights. This algorithm is similar to that in [21] which looks at different distributions of rates.

This algorithm assumes distortions are additive across multiple dropped packets, but allows for simple precedence constraints that can be depicted as trees. A precedence constraint of packet k to packet j means that packet k must precede packet j in the schedule. That is, packet j cannot be decoded correctly without the inclusion of packet k. Precedence constraints can be represented by a simple tree structure where all parent nodes must precede their children. In JPEG2000, we can assume that across different tiles, resolutions, color-components, and precincts, packets are independent. However, within the same tile, resolution, color-component, and precinct, packets are dependent across quality layers. Distortion is additive across quality layers only if the preceding layers are also included. Fig. 5(a) shows the tree structure for the precedence constraint for JPEG2000 packets within the same tile, resolution, color-component, and precinct. In H.264/MPEG-4 SVC with hierarchial B frames, there is a clear dependency between B frames. We can map these dependencies to a tree structure where each frame's parent is the most junior parent, as in Fig. 5(b). The most junior parent is the lowest parent in the precedence tree. For instance, B_3 has 2 parents: B_2 and B_4. However, B_4 is a parent to B_2 so the precedence constraint of B_4 on B_3 is captured by a single precedence constraint of B_2 on B_3. We assume that all I and P frames are successfully transmitted and received–I and P frames can be scheduled first and transmission only occurs if there is enough bandwidth to ensure successful reception of them all. Therefore, the only relevant precedence constraints occur between B frames. Finding the Rate-Distortion optimal subset of packets with these precedence structures is an instance of the Precedence Constraint Knapsack Problem [24], which can be solved optimally using dynamic programming.

While dynamic programming will give the optimal subset of packets given a rate constraint, the schedules are not embedded. Therefore, we proposed a fused-greedy algorithm to generate embedded schedules. This algorithm can be shown to give the optimal embedded schedule [21]. The algorithm

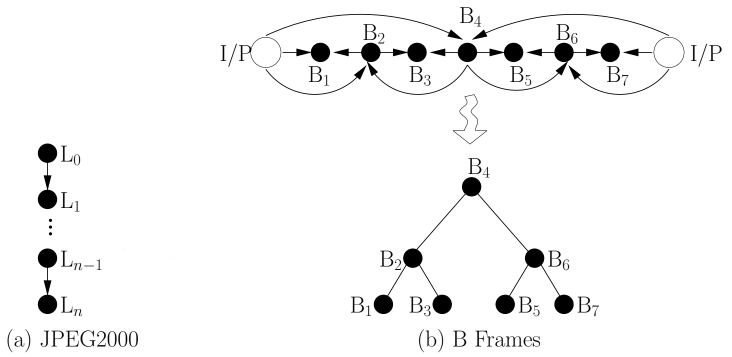


Fig. 5. Precedence constraints for JPEG2000 packets and SVC hierarchical B frames. For JPEG2000, within the same Tile, Resolution, Color Component, and Precinct, packets are dependent in a linear fashion. Packets corresponding to Layer i must be included prior to packets $j > i$. For hierarchical B frames, the hollow circles correspond to anchor I or P frames. The dependency structure can be mapped to a simple tree structure where each frame's parent is the most junior parent. We assume that all I/P frames are successfully transmitted and received, and therefore the dependencies on I/P frames are not listed.

takes in the distortion values, d_i, and sizes, s_i of each packet and returns the embedded schedule. Let $k_i = \frac{d_i}{s_i}$, be the distortion-to-size ratio. Let $\mathcal{P}$ be a set of integer pairs which represents the set of precedence constraints. If $(i, j) \in \mathcal{P}$, then packet j must precede packet i. The fused-greedy algorithm is as follows:

```
FUSED-GREEDY($\mathbf{d}, \mathbf{s}$)
1  $k_i = \frac{d_i}{s_i}, \forall i$
2  Check precedence constraints $\forall (i, j) \in \mathcal{P}$
3  **if** $k_i > k_j$: Violation between packet $i$ and $j$
4     **then** Fuse packets:
5           $k_i = \frac{d_{i,j}}{s_{i,j}} = \frac{d_i + d_j}{s_i + s_j}$
6           $k_j = 0$
7  Sort packets in descending order by $k_i$
```

Fusing packets i and j corresponds to generating a virtual packet consisting of packets i and j. This packet has a new distortion value and size equal to the sum of the 2 packets fused within it. By fusing packets i and j, an empty virtual packet is left behind. Note that fusing does not affect the contents of the packets; rather it serves as a way to view packets when making scheduling decisions. Certainly, fused packets can be separated during truncation and viewed as unique packets as long as the precedence constraints are satisfied.

The fused-greedy algorithm we have briefly described generates optimal embedded schedules for a single distortion measure; however, the extension to packets with multiple distortion measures is not immediate. As we further the discussion of Multiple Distortion Measures, we will continue to reference this algorithm. Furthermore, in Sections IV and V, we develop MDM-aware embedded scheduling algorithms which stem from the fused-greedy algorithm.

III. Spatial Resolutions: An Insightful Instance of Multiple Distortion Measures

In this section, we demonstrate, through quantitative and qualitative results, some of the gains that can be achieved by accounting for Multiple Distortion Measures. Specifically, we look at an insightful instance of Multiple Distortion measures as applied to multiple–low and high–display resolutions for images. By accounting for Multiple Distortion Measures when making scheduling decisions, up to 4dB gains can be achieved, as well as noticeable subjective improvements in image quality. In Section V, we show these gains in the case of temporal scalability–low and high frame rates–for video.

We consider the case where we have low and high resolution viewers. This would be the case if some users wish to view the content on a cellphone or PDA and others wish to view it on a laptop. We examine this scenario in the context of JPEG2000 encoded images and gain insight into the value of Multiple Distortion Measures.

In our experiments, the high resolution benchmark is the original image, $\mathcal{T}_H(X) = X$, and the low resolution benchmark, $\mathcal{T}_L(X)$, is a 4×4 downsampled version of the original high resolution image. In some cases, the original image may not be available, so the benchmark image $\mathcal{T}_H(X)$ would be the decoded image at the original high resolution. JPEG2000 is a packetized scalable image coding standard, where subsets of packets are independently decodable. To calculate the distortion values associated with each media packet, we incrementally drop packets along the dependency structure, decode, and calculate the resulting mean-squared error (MSE). Instead of comparing the decoded image to just the original high resolution image, $X = \mathcal{T}_H(X)$, we also compare to the low resolution benchmark image, $\mathcal{T}_L(X)$. Therefore, each packet has multiple (2) distortion values associated with it: one for each resolution.

A. Transformation: Spatial Downsampling

The downsampling captured by transformation, $\mathcal{T}_i$ can be done via one of the many different downsampling methods which exist. It is important to note that the rest of our results and analysis are independent of the downsampling method and only utilize the fact that multiple distortion values exist. In our experiments, we examine two linear methods for 2×2 downsampling: a basic block filter to do 2×2 pixel-averaging as well as the 13-tap downsampling filter developed by the Scalable Video Coding effort, which we denote by "SVC". We apply each 2×2 downsampling filter twice in order to achieve 4×4 downsampling.

Image	Rate	LowRes PSNR		opt PSNR	
	(bytes)	Pix-Avg	SVC	Pix-Avg	SVC
Actor	20386	30.52	36.43	32.72	36.70
Aerial	16344	32.22	37.97	34.46	38.74
Barboo	16119	28.85	34.23	29.81	34.28
Bike	17583	28.10	34.24	31.36	35.27
Cafe	17270	25.38	31.09	26.76	31.24
Woman	16983	36.21	41.10	38.71	41.47

Fig. 6. Spatial Scalability: Comparison of PSNR for 6 images when including all of the low resolution packets defined by the wavelet decomposition as compared to selecting the optimal packets at the same rate. Downsampling is done using pixel-averaging and the SVC downsampling filter.

Many scalable coders allow for images to be scaled down by resolution. Suppose one wanted to reduce the rate of the encoded bitstream with the goal of minimizing distortion of the low resolution image. This would result in selecting the JPEG2000 packets which minimize the resulting low resolution distortion. Typically, as in the case of the JPEG2000 codec, the resolutions are determined by the wavelet decomposition. It is therefore possible to reconstruct a low resolution image (downsample by $2^k \times 2^k$, $k \in \mathbb{N}$) by extracting only the packets which correspond to the low resolution wavelet packets. This would be identical to downsampling via the low resolution wavelet filter. However, while the wavelet decomposition is very effective for compression, it does not necessarily correspond to the most visually appealing low resolution version of the image. Also, by allowing for other downsampling methods, we generalize to user types where downsampling does not correspond to projecting onto a subspace defined by the wavelet filter. Because we obtain the low resolution image by downsampling the image via some method other than by the wavelet decomposition, often times the high resolution packets improve the low resolution image more than the low resolution packets. It is particularly surprising to see how much gain can be achieved by considering the high resolution packets. Table 6 summarizes comparisons of PSNRs evaluated for low resolution viewing for 6 different standard test images for JPEG2000. The benchmark images against which distortion is calculated are the downsampled images via pixel-averaging and using the H.264/MPEG-4 SVC filter. We first examine the PSNR when the image is reconstructed using all of the low resolution packets as defined by the wavelet decomposition. We compare this to an image reconstructed using packets optimally selected to minimize distortion given the same rate constraint. By allowing the selection of non-low resolution packets, we have $1-2$dB gains when using pixel-averaging, and $0-1$dB gains when using the H.264/MPEG-4 SVC downsampling filter. Clearly, if downsampling were done via the wavelet decomposition filter, there would be no gap.

B. Differences in Optimal RD Tradeoff

Scalable media allows for adaptation for various user types; however, as shown in the previous section, this is not always optimal. Another drawback to relying solely on the levels of scalability defined by the scalable codec is that it only defines a coarse granularity for scalability. For instance, a low resolution user type may have a rate constraint that does not allow for all low resolution packets to be transmitted. In this case, which packets should be discarded to generate the optimal Rate-Distortion tradeoff?

Again, define a schedule as the operator which, given a rate constraint and a set of packets, generates a subset of packets which adhere to the rate constraint. The *optimal* is then the subset of packets which minimizes distortion while adhering to the rate constraint. Typically, schedules are optimized for high resolution viewing even when the viewer has a low resolution display. We can see in Fig. 7 the PSNR versus Rate curves for schedules optimized at low and high resolutions. The solid lines correspond to the schedules optimized and evaluated at the low and high resolution distortion measures. The dashed curves correspond to the schedules optimized to minimize the low resolution distortion measure, but measured at the high resolution distortion measure, and optimized for high viewing, but measured at low. There is up to a 2dB gain in the low resolution PSNR when the packet selection is optimized for low resolution viewing rather than the original high resolution viewing and downsampling is done via pixel-averaging. This gain increases to 4dB when using the H.264/MPEG-4 SVC downsampling filter.

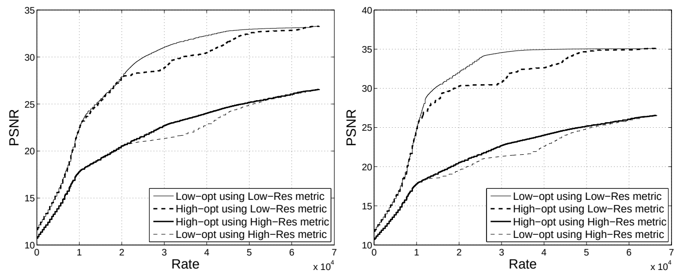


Fig. 7. Spatial Scalability: PSNR vs Rate (in bytes) for schedules optimized for low and high resolutions measured at both low and high resolution metrics. Thicker lines correspond to the performance of the High-Optimal Schedule. Downsampling by pixel-averaging (left) and SVC filter (right)

Fig. 8 provides a visual example of the benefits of Multiple Distortion Measures. Here, images are decoded at one-fifth the original bitrate. The image on the right is optimized for high resolution viewing and the image on the left for low. There is noticeable color degradation in the image on the right. Many of the cafe patrons in the middle of the image as well as detailing on the buildings have lost their color content. This is because when the image is optimized for high resolution viewing, edges become more important. So, edges are much more well defined for full resolution viewing on the right. However, once the image is reduced in size for low resolution viewing, these edges cannot be displayed in such a pronounced manner and are no longer as important. Therefore, bytes have been wasted on edges that cannot be seen on a low resolution display rather than on improving the color quality of the low resolution image. This is a key factor about why accounting for multiple distortion measures is important.

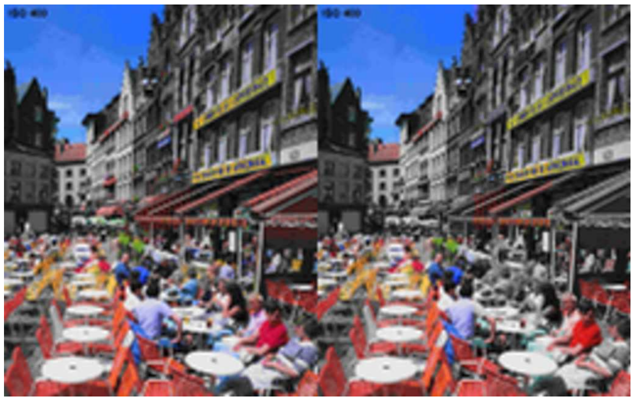


Fig. 8. Spatial Scalability: Decoded images at one-fifth of the original bitrate. The image on the left is optimized for low resolution viewing and the image on the right is optimized for high resolution viewing. Each image is decoded to the high resolution size of 512×640 pixels, then downsampled, using pixel-averaging, for low resolution viewing at 128×160 pixels. The image on the right is missing color quality for some of the cafe patrons in the middle of the image as well as on the building sides. This is due to the inclusion of high resolution detail, such as sharp edges, which cannot be displayed at low resolution viewing.

C. Correlation between Schedules

By examining some properties of schedules optimized for different distortion measures, we can gain some important insight into the causes for the drop in performance when optimizing for the wrong measure. Define a schedule $S_u(R)$ as the optimal subset of packets which minimize distortion according to distortion measure defined by $\mathcal{T}_u(\cdot)$ and given rate constraint R.

Define the correlation between the low and high resolution optimized schedules, S_L and S_H, as the fraction of packets from the low resolution schedule that are also in the high resolution schedule at the same rate constraint. Therefore $C(S_L, S_H) = \frac{|S_L \cap S_H|}{|S_L|}$. Correlation is a good way to measure the similarities between schedules. Fig. 9, shows the correlation between schedules optimized for different resolutions is fairly varied. At rates $25 - 45$ kbytes, the correlation is very low, which means the optimal packet selection for low and high resolution viewing is very disparate. This large discrepancy between schedules is why there are the PSNR gaps in Fig. 7 around the same rates. At very high and very low rates, the schedules are quite correlated. Clearly, at high rates, most packets are included in the schedule and the few that are not are negligible for both types of users. Also, at low rates, so few packets are selected that the same packets will create the foundation for the image, regardless of the viewing resolution.

It is quite surprising how different the low and high resolution optimized schedules are from each other. Because the two prioritizations of the packets differ so greatly, it is actually *impossible* for a single embedded schedule to jointly minimize distortion for the low and high resolution users. Therefore optimized PSNR vs. Rate performance in the previous section acts as an upperbound for any schedule.

In this section, we examined the use of Multiple Distortion Measures in the context of multiple viewing resolutions. By

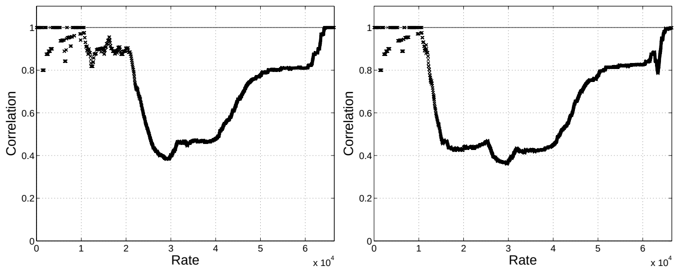


Fig. 9. Spatial Scalability: Correlation vs Rate (in bytes) between schedules optimized for low and high resolution viewing. Downsampling by pixel-averaging (left) and SVC filter (right)

taking MDM into consideration, up to 4dB gains can be achieved with spatial scalability. In Section V, we look at the gains in the context of temporal scalability. Systems which incorporate Multiple Distortion Measures need to determine how to generate schedules given these multiple measures. We examine this question in the rest of the paper.

IV. An Algorithm for Embedded Scheduling with Multiple Distortion Measures

To examine the benefits of considering Multiple Distortion Measures when making scheduling decisions, we examine an instance with U user types. We turn our focus to a special, insightful case where rates are uniformly distributed along the non-overlapping support for each user type. Therefore, at each rate, the distortion is measured using only one distortion measure. The goal is to generate an embedded schedule which minimizes the expected distortion. In this case, there are only U transformation operators and U associated distortion lists in Fig. 4.

We consider the scenario where we switch between distortion measures at distinct switching rate $R_s(u)$: i.e. for rates $0 \leq R < R_s(1)$, all distortion is measured according to $\mathcal{T}_1$, for rates $R_s(1) \leq R < R_s(2)$, all distortion is measured according to $\mathcal{T}_2$, and for all rates $R_s(u-1) \leq R < R_s(u)$, all distortion is measured according to $\mathcal{T}_u$. We assume there is a uniform distribution of rates, so that with probability $\frac{1}{R_{max}}$, a user will view the content at rate R. This is a special case of a more general distribution of user types and rate constraints.

Even with 2 user types (a single switching rate, R_s), there is a conflict between objectives: minimizing $E[D_1]$ versus minimizing $E[D_2]$. In fact, the vast discrepancies, even in this simple scenario, are surprising and help validate the need to incorporate Multiple Distortion Measures. One way to examine this conflict is to look at the similarities and disparities between packet selection for each schedule. Suppose that the low rate users wish to view an image at low resolution, so $\mathcal{T}_1(X)$ corresponds to downsampling. Also suppose that the high rate users wish to view the image at the original high resolution so that $\mathcal{T}_2(X) = \mathcal{T}_I(X) = X$. For simplicity, assume that the switching rate is half the bitrate of the entire image, $R_s = \frac{R_{\max}}{2}$. In order to understand the (dis)similarities of packet rank/importance across the two distortion measures, we examine the packets chosen before and after R_s to see how many are similar. If the low and high resolution optimized

Image	Total # of bytes	Frac. bytes before R_s		Frac. bytes after R_s	
		Pix-Avg	SVC	Pix-Avg	SVC
Actor	64046	0.3661	0.3615	0.3683	0.3569
Aerial	52514	0.3762	0.3706	0.3811	0.3777
Barboo	51567	0.3674	0.3607	0.3678	0.3627
Bike	65736	0.3937	0.3439	0.4010	0.3450
Cafe	66477	0.3544	0.3473	0.3513	0.3547
Woman	66216	0.3126	0.2989	0.3301	0.3227

Fig. 10. Spatial Scalability: Fraction of total packets, measured by their size in bytes, that are common to both the low, S_L, and high, S_H, resolution optimized schedules before and after the switching rate, R_s. If the schedules were identical, $S_L = S_H$, then the fraction of bytes before and after R_s would be .5 since $R_s = \frac{R_{\max}}{2}$.

Image	Total # of pkts	Frac. pkts before R_s		Frac. pkts after R_s	
		Pix-Avg	SVC	Pix-Avg	SVC
Actor	540	0.2426	0.2392	0.4574	0.4527
Aerial	432	0.1740	0.1717	0.5940	0.5777
Barboo	432	0.1921	0.1852	0.5417	0.5255
Bike	540	0.2338	0.2245	0.5436	0.4842
Cafe	540	0.2189	0.2134	0.4787	0.4378
Woman	540	0.2356	0.2352	0.4601	0.4426

Fig. 11. Spatial Scalability: Fraction of total packets that are common to both the low, S_L, and high, S_H, resolution optimized schedules before and after the switching rate, R_s.

schedules were equal, $S_1 = S_2$, then all the packets would be identical. In this case, half of the total number of bytes in the bitstream would be prior to R_s and half would be after. Table 10 shows that approximately 35% of the bytes are common before and after R_s. It is the discrepancy of the remaining 30% of the bitstream which causes the significant drops in PSNR when optimizing for the wrong distortion measure, as shown in Fig. 7. Examining the discrepancies in terms of bitrate is more intuitive than looking at the discrepancies in terms of packets. However, scheduling is done on a packet-level basis, rather than on a bit/byte-level basis. Table 11 summarizes the fraction of total packets which are common between the low and high resolution schedules before and after the switching rate, R_s. A large percentage of packets are the same after the switching rate; however, their contribution, in terms of bitrate, is approximately 35%. For both the high and low resolution user, the least important packets tend to be small in size. While it may seem insignificant that the low and high resolution optimal schedules disagree on the importance of about 30% of the bitstream, it is these discrepancies which lead to loss in performance when ignoring Multiple Distortion Measures.

Acknowledging these competing objectives, we aim to find an embedded schedule to minimize the expected distortion. As defined in Section II, $D_u(S(R))$ is the distortion measured, with benchmark $\mathcal{T}_u(X)$, for schedule S evaluated at rate R. Let $d_{u,i}$ denote the distortion incurred by user type u if he does not receive packet i, excluding the additional distortion incurred due to the inability to decode all packets which packet i must precede. Equivalently, $d_{u,i}$ denotes the amount distortion is reduced if user type u receives packet i, assuming all packets preceding packet i have been received. A schedule, S, can also be defined by $\{r_i\}$, the rate at which packet i is included in the schedule. Then, given a schedule S, the expected distortion ($w_{u,R} = 1$) is given by:

$$\begin{aligned} E[D(S)] &= \frac{1}{R_{\max}}\left[\sum_{R=0}^{R_{max}}\sum_{u=1}^{U} D_u(S(R))\mathbf{1}_{\{R_s(u-1)\le R<R_s(u)\}}\right] \\ &= \frac{1}{R_{\max}}\left[\sum_{u=1}^{U}\sum_{R=R_s(u-1)}^{R_s(u)-1} D_u(S(R))\right] \\ &= \frac{1}{R_{\max}}\left[\sum_{u=1}^{U}\sum_{R=R_s(u-1)}^{R_s(u)-1}\left(\sum_i d_{u,i}\mathbf{1}_{\{R<r_i\}}\right)\right] \end{aligned} \quad (3)$$

where $\mathbf{1}_{\{A\}}$ is an indicator such that $\mathbf{1}_{\{A\}} = 1$ if A is true and 0 otherwise.

Define s_i as the size in bytes of packet i. As defined in Section II-C, $\mathcal{P}$ is the set of packets with precedence constraints between them. For instance, if packet i and j corresponded to B_1 and B_2, respectively, of the same GOP, then $(i,j) \in \mathcal{P}$, since B_2 precedes B_1, as seen in Fig. 5(b). And R_s is the switching rate at which we switch from distortion measure defined by $\mathcal{T}_L$, to distortion measure defined by $\mathcal{T}_H$. The optimization problem is to find schedule S^* over all possible schedules that satisfies:

$$\begin{aligned} \min_{S\in\mathcal{S}} \quad & \frac{1}{R_{\max}}\left[\sum_{u=1}^{U}\sum_{R=R_s(u-1)}^{R_s(u)-1}\left(\sum_i d_{u,i}\mathbf{1}_{\{R<r_i\}}\right)\right] \\ s.t. \quad & r_i \ge r_j,\ (i,j)\in\mathcal{P} \\ & \sum_i s_i\mathbf{1}_{\{r_i\le R\}} \le R,\ \forall R \end{aligned} \quad (4)$$

where $R_{\max}$ is the total number of bytes in the image. The first constraint corresponds to the precedence constraint. The second constraint is the rate capacity constraint.

The objective and last constraint of the optimization problem are nonlinear, which makes this problem hard. There are $n!$ possible permutations of n packets; hence $n!$ possible schedules. An exhaustive search of all possible schedules to find the minimum expected distortion would be computationally infeasible. We want to find a less computationally expensive algorithm that achieves high performance.

A. MDM-fused Scheduling Algorithm

We present an algorithm which runs in $\Theta(Un^U)$ by using what we call a changing-rate. The basis of this algorithm is the fused-greedy algorithm of Section II-C, which we unite with Multiple Distortion Measures. We refer to this algorithm as the ”MDM-fused” algorithm.

We relay the key idea behind this algorithm by focusing on the case of $U = 2$ users types: L and H. Intuitively, for small R_s all packets should be prioritized based on their low rate distortion measure, $d_{L,i}$. Likewise, for high R_s, all packets scheduled are prioritized based on their high rate distortion measures, $d_{H,i}$. For intermediate values of R_s, there should be

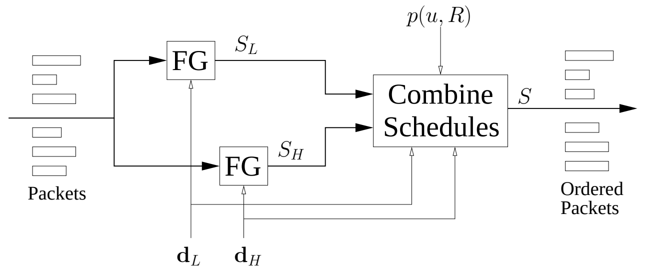


Fig. 12. Block diagram of our MDM-aware schedule. The fused-greedy algorithm is performed on the low and high distortion metric to generate low and high optimal schedules, S_L and S_H. The combining of schedules can be done in multiple ways. We propose to fix a changing rate to change between the optimal low schedules, S_L, to the optimal high schedules, S_H. One can also iterate over changing rates to find the optimal one, R_c^*.

a balance between the high, S_H, and low, S_L, schedules. Fig. 12 is a block diagram of an algorithm of this nature where the "Combine Schedule" block specifies how to balance these two schedules. We propose an algorithm which utilizes a changing-rate, R_c, to define this block. R_c is the rate at which we change from low to high rate scheduling, i.e., for $R \leq R_c$ packets are scheduled according to its low rate distortion measure, $d_{L,i}$, and for $R > R_c$ the remaining packets are scheduled according to its high rate distortion measure, $d_{H,i}$. The changing rate allows the schedule to switch from focusing on low to high rate users. Given a switching rate R_c, packet distortion information $d_{L,i}$ and $d_{H,i}$, and packet sizes s_i, the changing-rate scheduling algorithm is (Note that steps 3 and 4 make up the "Combine Schedule" block in Fig. 12):

```
CHANGING-RATE-SCHEDULING($\mathbf{d}_L, \mathbf{d}_H, \mathbf{s}, R_c$)
1  $S_L$ = FUSED-GREEDY($\mathbf{d}_L, \mathbf{s}$)
2  $S_H$ = FUSED-GREEDY($\mathbf{d}_H, \mathbf{s}$)
3  Fill $S$ according to $S_L$ until $S$ has $R_c$ bytes
4  Fill remaining packets into $S$ according to $S_H$
5  return $S$
```

In Fig. 13, the PSNR versus Rate curves for various R_c values are plotted where $\mathcal{T}_L$ corresponds to downsampling to a low resolution benchmark image and $\mathcal{T}_H$ corresponds to the original high resolution image. There is clearly a trade-off between the competing objectives of optimizing for low versus high resolution viewing. The curves vary significantly as the changing rate, R_c, varies for a fixed switching rate, $R_s = 33$ kbytes. For low values of R_c, packets are scheduled according to high resolution distortion measures starting at low rates. Therefore, the performance of the high resolution users improves significantly, while the performance of the low resolution users degrades significantly. Likewise, for large R_c, the low resolution performance is very high while the high resolution performance takes a hit.

Every value of R_c corresponds to a schedule whose expected distortion, $E[D]$ can be evaluated. Fig. 14 shows the expected distortion as a function of the changing rate, R_c, with a fixed switching rate, R_s. Empirically, there is a unique R_c that corresponds to the minimum expected distortion. This optimal changing rate, R_c^*, depends on the switching rate, R_s, as well as the packet distortion values and sizes, $\mathbf{d}_L$, $\mathbf{d}_H$, and $\mathbf{s}$. Note also that these results are for uniform probability distributions and uniform weights and the actual value for R_c^* will differ as these change. Given a coded sequence of media packets and a fixed switching rate, we can search over R_c to find R_c^* which minimizes the expected distortion.

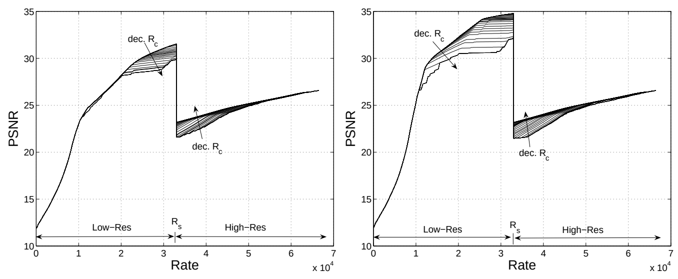


Fig. 13. Spatial Scalability: PSNR vs. Rate for varying values of the changing rate, R_c. $R_s = 33$ kbytes. Downsampling using pixel-averaging (left) and SVC filter (right).

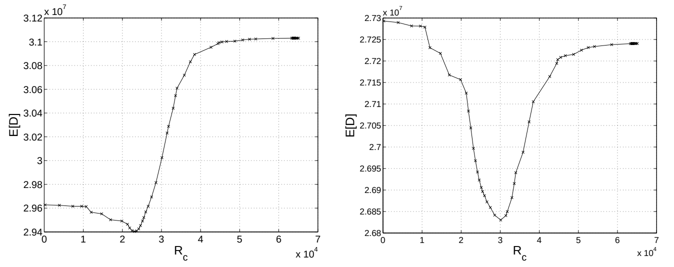


Fig. 14. Spatial Scalability: Expected distortion vs. changing rate with $R_s = 30$ kbytes (left) and $R_s = 40$ kbytes(right). Downsampling by pixel-averaging.

We can modify the changing-rate scheduling algorithm to incorporate the search to find the optimal R_c^*. We call this algorithm the "MDM-fused" algorithm as it is based on the fused-greedy algorithm, but it is MDM-aware by searching for R_c^* to optimize the tradeoff between users. Without loss of generality, index packets by the optimal low resolution schedule, S_L. Let R_i correspond to the rate at which packet i is included in S_L. Therefore, $R_1 = 0$, $R_2 = s_1$, $R_3 = s_1 + s_2$, etc. where s_i corresponds to the size of the i^{th} packet of S_L. We have just modified the "Combine Schedule" block in Fig. 12 to incorporate the search for the optimal R_c^* in steps 4 through 9 of the new scheduling algorithm:

```
MDM-FUSED($\mathbf{d}_L, \mathbf{d}_H, \mathbf{s}$)
1  $S_L$ = FUSED-GREEDY($\mathbf{d}_L, \mathbf{s}$). Set $S^* = S_L$
2  $S_H$ = FUSED-GREEDY($\mathbf{d}_H, \mathbf{s}$)
3  for $i \leftarrow 1$ to $n$
4     do $R_c \leftarrow R_i$
5          Fill $S$ according to $S_L$ until $S$ has $R_c$ bytes
6          Fill remaining packets into $S$ according to $S_H$
7          if $E[D|S] < E[D|S^*]$
8             then $S^* \leftarrow S$
9  return $S^*$
```

Thus far, this algorithm has focused on 2 user types, but the extension to more user types is trivial. Instead of employing a single changing rate between users 1 and 2, we require $U-1$ changing rates–one ($R_c(1)$) between users 1 and 2, ($R_c(2)$) users 2 and 3, etc.

1) Complexity Analysis: Here we analyze the run-time of the changing-rate algorithm with search for R_c^*. The search space has been reduced from an exhaustive search over all $n!$ possible schedules to a special subset of n^{U-1} schedules. Each schedule takes $\Theta(n)$ time to evaluate, which gives a total run-time of $\Theta(Un^U)$.

There are n total media packets. The fused-greedy algorithm takes $\Theta(n \log n)$ to find the optimal schedules for each user type, S_u. To generate the schedule for a given set of changing rates, we incrementally add packets from S_u until rate $R_c(u)$. Then we scan through S_{u+1} and add remaining packets that have not yet been added until rate $R_c(u+1)$. This process takes $\Theta(Un)$ to generate the resulting schedule as we step through S_u packet by packet. The expected distortion is a summation of n terms corresponding to the expected distortion contribution of each of the n packets. Therefore, it takes $\Theta(n)$ to calculate the expected distortion. For each switching rate, there are n distinct changing rates which will give different schedules: one after the first packet in S_u, after the second packet in S_u, after the third packet in S_u and so on. Therefore, to evaluate the expected distortion for all n^{U-1} changing rates takes $\Theta((U+1)n^U)$. We can find the best changing rate in linear time while we evaluate the expected distortion. This gives a total run time of $\Theta((U+1)n^U + Un \log n) = \Theta(Un^U)$. This can be costly for a large number of user types, but is very manageable for 3 or less types. Even considering just 2 user types will prove to have large gains.

B. MDM-switch Heuristic

For a very large number of user types, MDM-fused can be quite computationally intensive. Rather than searching for the optimal R_c^*, another option is to set $R_c = R_s$. This is a natural heuristic with complexity $\Theta(Un \log n)$ since there is only one changing rate per switching rate. We call this scheduling algorithm the ”MDM-switch” algorithm since the changing rate is equal to the switching rate. This policy is MDM-aware in the sense that it tries to balance between the optimal schedules, given by the fused-greedy algorithm, for each of the U user types. However, it is easy to see that MDM-fused will perform better than MDM-switch as $R_c = R_s$ is a possible solution to R_c^*. As we will see in the following discussion, in some cases the gap in performance will be significant, whereas in others, it is minimal.

C. Results

In this section we present results for the performance of MDM-fused and the MDM-switch heuristic in the context of $U = 2$ user types. We compare the performance of this algorithm to the conventional approach which generates schedules using the fused-greedy algorithm of Section II-C assuming a single distortion measure of the high resolution measure. For completeness, we also compare to the fused-greedy algorithm performed on just the low resolution distortion measure.

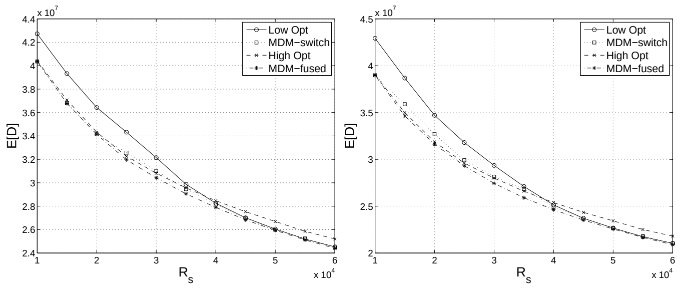


Fig. 15. Spatial Scalability: Expected distortion vs. switching rate for various algorithms. Downsampling by pixel-averaging (left) and SVC filter (right).

In each algorithm, schedules are determined based on empirically calculated distortion values which can be stored in the packet headers as in [25]. Distortion is assumed to be additive across multiple packet drops. If a precedence constraint is violated, i.e., a packet is included but a packet corresponding to its lower quality layer is not, the packet's inclusion does not reduce distortion. While all schedules are generated according to this model, their performance is evaluated via decoded images. We present the results for the Cafe image, although the trends and performance gains are similar for the other images.

Fig. 15 shows the expected distortion, $E[D]$, versus the switching rate, R_s. For low switching rates, the high resolution optimal schedule, S_H, performs very well. This is because the high and low schedules are nearly identical at low rates, which results in little loss in performance for the low resolution viewers, and optimal performance to the majority of users who are high resolution viewers. However, as R_s increases, the performance of the high resolution schedule drops because it ignores the low resolution users and their different distortion metric. Likewise, the low resolution schedule performs well for high R_s, but very poorly for low R_s. Setting $R_c = R_s$ can outperform the low and high resolution schedules because it tries to account for multiple distortion measures by switching between the low and high distortion metrics. However, we can see that if we optimize R_c, we can achieve even higher performance. MDM-switch may be more favorable in situations with many user types as the complexity of MDM-fused may limit is practicality. However, MDM-switch is easy and quick to implement, while out performing scheduling algorithms based on one distortion measure.

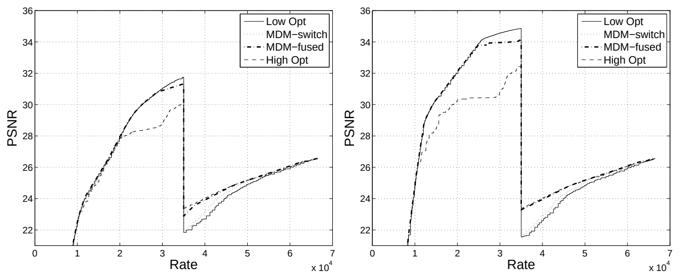


Fig. 16. Spatial Scalability: PSNR versus rate with $R_s = 35$ kbytes for various algorithms. Downsampling by pixel-averaging (left) and SVC filter (right).

Fig. 16 shows the PSNR versus Rate curves for the different scheduling algorithms given switching rate, $R_s = 35$ kbytes. The MDM-switch policy ignores the high resolution schedule for too long, and switches to S_H much too late. Therefore, its performance is quite similar to the low optimized schedule. However, because it does account for the high resolution user for rates above R_s, it outperforms the optimal low resolution schedule, S_L, which completely ignores all high resolution users. The proposed MDM-fused policy clearly outperforms the others. There is a very slight drop in PSNR, at most .1dB, from the high optimal schedule just after the switching rate. However, at rates $R < R_s$, MDM-fused performs nearly 2dB better than the standard approach of the high optimized schedule, and nearly as well as the low optimized schedule. When downsampling using the H.264/MPEG-4 SVC filter, over 3dB gains are achieved.

Intuitively, as the switching rate increases, so will the changing rate. It is interesting to note that $R_c^* \leq R_s$, since once the user type switches, there is no benefit to scheduling according to the incorrect distortion measure. Thus far, we have assumed a uniform distribution of rates at which users will consume the media. As we increase the weight for the low resolution user ($w_{L,R}$), or equivalently, increase the probability of a low resolution user ($p(L, R)$), the optimal changing rate, R_c^* will increase. By increasing the weight or probability of the low resolution users, the performance of the low resolution users contributes more to the expected weighted distortion. Hence, the performance of the low resolution user is more important, and that of the high resolution user is sacrificed by changing to high resolution scheduling at a later rate. Conversely, if the weight, or probability of the high resolution users were increased, R_c^* would decrease. Certainly, R_c^* depends on the distribution of user types as will the performance of the different scheduling algorithms. The question of how to schedule with arbitrary distributions of user types remains an interesting research problem which we are currently exploring.

V. TEMPORAL SCALABILITY

Multiple Distortion Measures can be applied in a number of settings. Thus far, we have presented experimental results for the case of still images and spatial scalability. To emphasize the generality of the Multiple Distortion Measures framework, we now present experimental results of MDM for temporal scalability in the case of video encoded using H.264/MPEG-4 SVC[1]. An example of such a scenario is two mobile devices, where one is very power constrained and thus chooses to consume the video at a lower frame rate, while the other prefers the highest quality and chooses to consume the video at the original high frame rate. SVC has spatial, temporal, and quality scalability. We focus on temporal scalability to highlight the gains that can be achieved when accounting for users with different frame rates; however, we stress that MDM can be used in conjunction with multiple forms of scalability, including a combination of spatial and temporal scalability.

[1]Because we are only using the temporal scalability of SVC, this video is also compatible with H.264/MPEG-4 AVC.

In this scenario, we wish to transmit 240 frames of the Soccer sequence in CIF format with an original frame rate of 30 frames per second. This sequence has periods of low background motion and minor foreground motion as well as periods with large background and foreground motion. We encode using a GOP structure of 8 frames and an intra-refresh every 16 frames. In Fig. 5(b), we map the dependencies of the hierarchical B frames into a tree capturing the precedence constraints. We only allow rate reduction and scalability by discarding B frames, so we assume all I and P frames are successfully transmitted and received. If a frame is dropped, we use frame copy error concealment to reconstruct the missing frame.

Suppose there are 3 users types. One type of user wishes to view the video at the original high frame rate of 30 frames per second, another type wishes to view the video at a lower frame rate of 15 frames per second, and the final type wishes to view the video at the lowest frame rate of 7.5 frames per second. In this scenario, the transformation operation is a straightforward frame dropping operation. The distortion measure for the 30Hz user is the standard average MSE per frame compared to the original sequence. The distortion measure for the 15Hz user is the average MSE per frame compared to every other frame in the original sequence–the original sequence temporally downsampled by a factor of 2. Likewise, the distortion measure for the 7.5Hz user is the average MSE per frame compared to every forth frame of the original sequence–a temporal downsampling by a factor of 4.

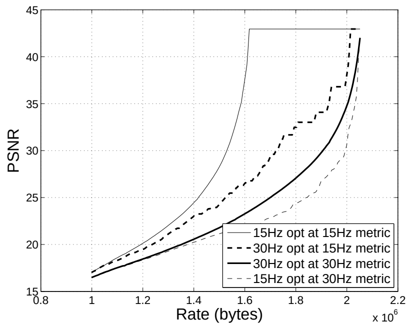


Fig. 17. Temporal Scalability: PSNR vs Rate (in bytes) for schedules optimized for viewing at 15 frames per second and 30 frames per second measured at both frame rate metrics. Thicker lines correspond to performance of the schedule optimized for 30 frames per second.

Fig. 17 shows the PSNR versus Rate curves for embedded schedules optimized for frame rates of 30Hz and 15Hz. This is the analogous figure to Fig. 7 for temporal scalability. The two solid lines correspond to the schedules which are both optimized and evaluated at the same frame rate distortion measures, e.g., optimized at 15Hz and evaluated at 15Hz. The dotted curves correspond to the schedules optimized to minimize the 15Hz distortion measure, but measured at the 30Hz distortion measure, and optimized for 30Hz viewing, but measured at 15Hz. There is a 12dB improvement in the 15Hz frame rate PSNR when the frame selection is optimized for 15Hz viewing rather than the original 30Hz viewing. All of the odd numbered B frames in the original 30Hz frame

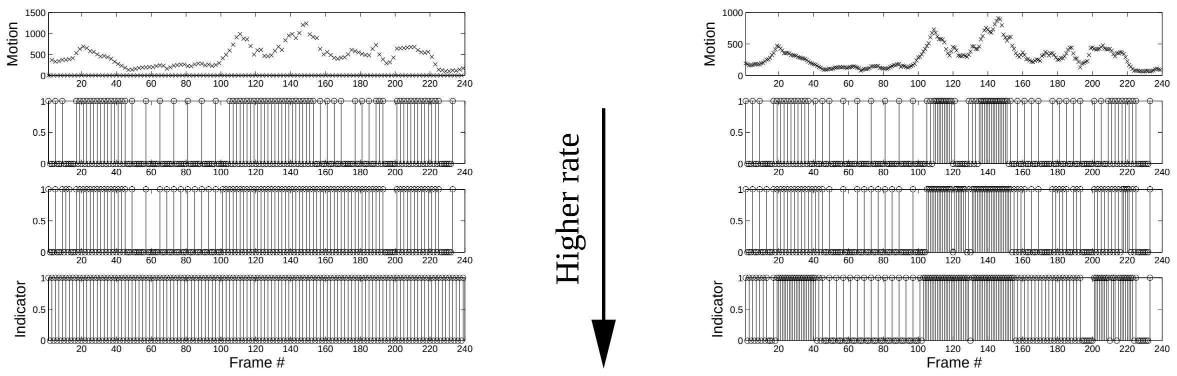


Fig. 18. Temporal Scalability: Motion between subsequent frames, measured by mean-squared error between adjacent frames. Motion $= E[|F_i - F_{i-1}|^2]$ (top). Frame inclusion for a given bit-rate constraint optimized for 15 frames per second (left) and 30 frames per second (right) viewing. The bit-rate constraint is constant for each horizontal plot (between 15Hz and 30Hz) and increases vertically. When the bit-rate constraint does not allow all packets to be transmitted, the frames corresponding to low motion in the video sequence are the first to be dropped.

rate sequence are dropped during the temporal downsampling to generate the 15Hz frame rate sequence. As a result, these frames have zero associated distortion to a 15Hz user and the transmission of these frames cannot improve the PSNR of the 15Hz video sequence. However, if these frames correspond to a section of the video with very high motion, they may be very important (large associated distortion) to a user viewing the content at 30Hz. This vast discrepancy in distortion values of the same frames across multiple users results in the multiple dB gains in PSNR when optimizing transmission specifically for the correct user type. Analogous results hold for embedded schedules optimized for 7.5Hz frame rates.

Frame selection is directly correlated with the amount and temporal location of motion in the video sequence. Fig. 18 shows just how dependent the frame selection is on motion. We express the amount of motion by the per-pixel mean-squared error between adjacent frames. Therefore, the motion at frame i is Motion$(i) = E[|F_i - F_{i-1}|^2]$, where F_i corresponds to frame i. The top plot corresponds to the amount of motion in the original sequence at 15Hz and 30Hz as a function of the original frame number. Because the 15Hz video is an integer downsample of the original, every other frame has 0 motion. The 3 other plots correspond to the frame selection given increasing bitrate constraints which are identical for each user type. For low bitrates, not all frames can be transmitted. The ones that are transmitted correspond to sections of large motion in the video sequence. As the rate constraint increases, more frames can be added. For the bottom plot, the 15Hz user receives *all* packets which benefit him. However, his packet selection is quite different from that of the 30Hz user. Instead of receiving frames at a periodic rate, it is optimal for the 30Hz user to lose some frames which correspond to low motion areas in order to transmit at a higher frame rate in high motion areas.

Clearly, the optimal frame selection for users with different frame rates conflicts with each other. We apply the changing-rate scheduling algorithm to the case of temporal scalability. In [22], the authors found that there exists distinct switching rates at which the preferred frame rate changes, which bolsters the validity of this type of scheduling scenario. In this case, we assume $\mathcal{T}_3$ and $\mathcal{T}_2$ correspond to the temporally downsampled benchmark video to be viewed at 7.5 and 15 frames per second, respectively, and $\mathcal{T}_1$ is the original video sequence to be viewed at 30 frames per second. Now that there are 3 levels of scalability, we have 2 switching rates–one between 7.5Hz and 15Hz viewers and another between 15Hz and 7.5Hz users. So for rates, $R < R_s(1)$ all users view the video at 7.5 frames per second; for $R_s(1) \leq R < R_s(2)$, all users view at 15 frames per second; and for $R_s(2) \leq R$, all user view at 30 frames per second. Fig. 19 shows the expected distortion as a function of $R_s(2)$, the switching rate from 15 to 30Hz viewing, for 2 different values of $R_s(1)$, the switching rate from 7.5 to 15Hz viewing. Again varying the switching rates modifies the relative performances of the single distortion measure schedules (30Hz opt, 15Hz opt, and 7.5Hz opt). In some cases, MDM-switch performs identically to MDM-fused (as when $R_s(1) = 1200$kbytes). As more user types are considered, this may prove to be a very effective and efficient scheduling algorithm. However, we can see that optimizing for the best changing rate can vastly improve performance (as when $R_s(2) = 1600$kbytes). In the case of temporal scalability, we see that accounting for Multiple Distortion Measures has a significant impact.

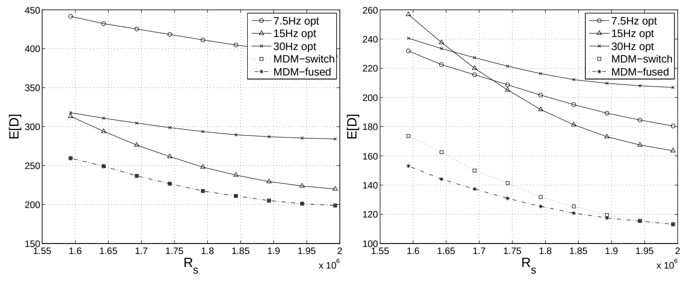


Fig. 19. Temporal Scalability: Expected distortion vs. switching rate $R_s(2)$ (15Hz to 30Hz) for various algorithms. For switching rate $R_s(1)$ (7.5Hz to 15Hz) of 1200 (left) and 1600 (right) Kbytes.

In Fig. 20, we plot the PSNR versus Rate curves for fixed switching rates of $R_s(1) = 1260$kbytes and $R_s(2) = 1600$kbytes as well as the Distortion versus Rate curves.

We can see that with little loss in performance prior to the switching rates, our scheduling algorithm, MDM-fused, is more than 10dB better than the conventional approach of optimizing for 30Hz viewing. Note that the optimization goal is to minimize the expected distortion. PSNR is the standard objective metric for evaluating the performance of video systems, so we also present the results in terms of PSNR. However, because we are minimizing the expected distortion, and because of the nonlinear mapping to PSNR, the PSNR can be somewhat misleading. By looking at the results in terms of distortion, one can see our policy successfully balances the tradeoff between 30Hz, 15Hz, and 7.5Hz viewing and nearly achieves the optimal distortion for all rates.

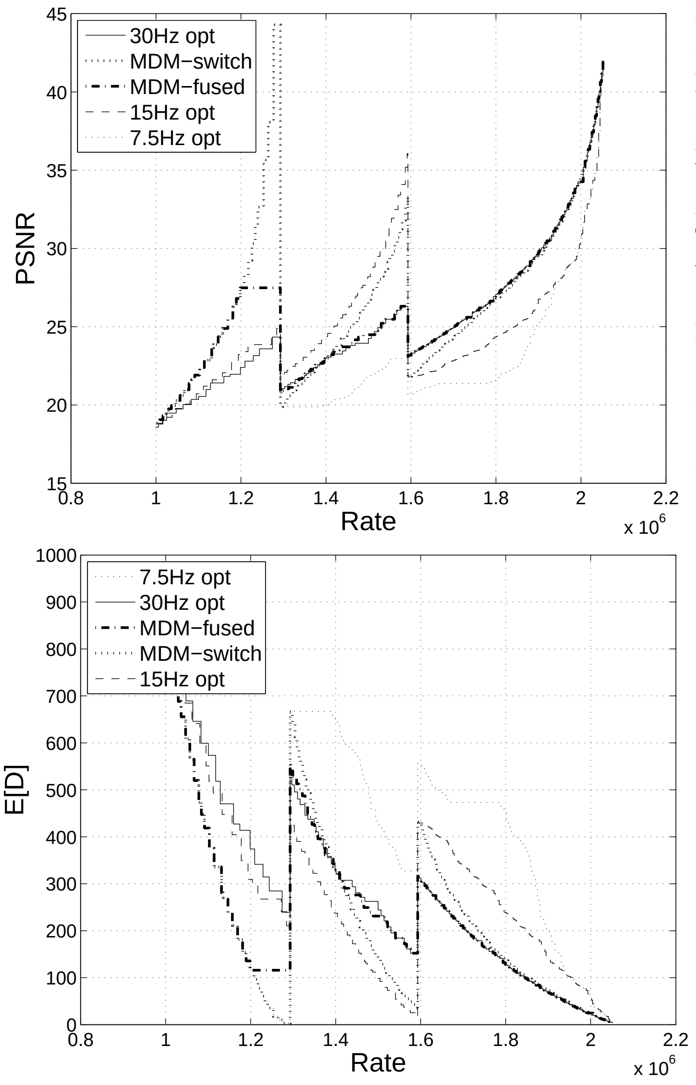


Fig. 20. Temporal Scalability: PSNR (top) and Distortion (bottom) versus Rate with $R_s(1) = 1260$kbytes and $R_s(2) = 1600$kbytes for various algorithms.

VI. Conclusions

In this paper, we presented a new framework to evaluate the performance of multimedia systems which serve multiple types of users. With the growing diversity in multimedia users, Multiple Distortion Measures will allow service providers to efficiently serve multimedia streams in a manner that accounts for the various needs of each user. We showed that quality measures used for streaming media are highly dependent on user types. In fact, in the case of packetized media, a packet's importance can be quite different depending on which user consumes it. These differences cause conflicts when simultaneously scheduling media packets to multiple user types. We also presented a framework in which to evaluate embedded scheduling algorithms for systems with Multiple Distortion Measures. We developed an MDM-aware embedded scheduling algorithm based on our prior work which assumed only a single distortion measure. We applied our framework and scheduling algorithm to 2 cases where MDM is relevant: spatial scalability for various resolutions and temporal scalability for various frame rates. These examples are illustrative of the gains which can be achieved by accounting for MDM, but are by no means exhaustive. MDM is a general framework which can be applied to any type of benchmark images or videos. Spatial scalability was explored in the case of still images through the JPEG2000 standard. Temporal scalability was studied in the case of video streaming with H.264/MPEG-4 SVC. In our experiments, for spatial scalability with JPEG2000, accounting for Multiple Distortion Measures resulted in up to 4dB gains and for temporal scalability with H.264/MPEG-4 SVC, gains of up to 12dB. By accounting for the diverse needs of its clients, a multimedia server can significantly improve the provided service by using and accounting for Multiple Distortion Measures.

References

[1] *15444 JPEG-2000 Image Coding System, Part 1: Core coding system*.

[2] Joint Video Team of ISO/IEC MPEG & ITU-T VCEG, *AHG Report on Spatial Scalability Resampling, Document JVT-Q007*, Oct. 2005.

[3] E. C. Reed and J. S. Lim, "Optimal multidimensional bit-rate control for video communication," *IEEE Trans. Image Process.*, vol. 11, pp. 873–885, Aug. 2002.

[4] J. Lee and B. Dickinson, "Temporally adaptive motion interpolation exploiting temporal masking in visual perception," *IEEE Trans. Image Process.*, vol. 2, pp. 513–526, Sept. 1994.

[5] J. Chen and H. Hang, "Source model for transform video coder and its application II: Variable frame rate coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 299–311, April 1997.

[6] H. Song, J. Kim, and C. Kuo, "Improved H.263+ rate control via variable frame rate adjustment and hybrid I-frame coding," in *Proc. IEEE ICIP*, pp. 375–378, Oct. 1998.

[7] H. Song, J. Kim, and C. Kuo, "Real-time encoding frame rate control for H.263+ video over the internet," *Signal Processing: Image Communication*, vol. 15, pp. 127–148, September 1999.

[8] F. Ishtiaq and A. Katsaggelos, "A rate control method for H.263 temporal scalability," in *Proc. IEEE Int. Conf on Image Processing (ICIP '99)*, vol. 4, (Kobe, Japan), pp. 280–284, Oct. 1999.

[9] N. Bozinovic and J. Konrad, "Modeling motion for spatial scalability," in *Proc. IEEE ICASSP*, pp. 29–32, May 2006.

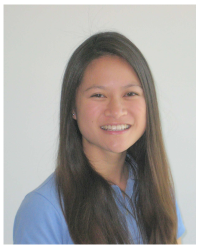
[10] H. Sun, W. Kwok, and J. Zdepski, "Architectures for mpeg compressed bitstream scaling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 191–199, Apr. 1996.

[11] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," in *Proc. IEEE Int. Conf on Acoustics, Speech, and Signal Processing (ICASSP '98)*, vol. 5, (Seattle, WA, USA), pp. 2813–2816, May 1998.

[12] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Process. Mag.*, vol. 20, pp. 18–29, Mar. 2003.

[13] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," *IEEE Trans. Multimedia*, vol. 2, pp. 101–110, June 2000.

[14] S. Wee, J. Apostolopoulos, and N. Feamster, “Field-to-frame transcoding with temporal and spatial downsampling,” in *Proc. IEEE Int. Conf on Image Processing (ICIP ’99)*, Oct. 1999.
[15] S. Wee and J. Apostolopoulos, “Secure scalable streaming and secure transcoding with JPEG-2000,” in *Proc. IEEE Int. Conf on Image Processing (ICIP ’03)*, vol. 1, pp. 205–208, Sept. 2003.
[16] M. Podolsky, S. McCanne, and M. Vetterli, “Soft arq for layered streaming media,” *Univ. California, Comput. Sci. Div., Berkeley, Tech. Rep. UCB/CSD-98-1024*, Nov. 1998.
[17] Z. Miao and A. Ortega, “Optimal scheduling for streaming of scalable media,” in *Proc. Asilomar Conf. Signals, Systems, and Computers*, vol. 2, (Pacific Grove, CA, USA), pp. 1357–1362, Nov. 2000.
[18] M. Kalman and B. Girod, “Techniques for improved rate-distortion optimized video streaming,” *ST Journal of Research*, vol. 2, pp. 45–54, Nov. 2005.
[19] P. Chou and Z. Miao, “Rate-distortion optimized streaming of packetized media,” *IEEE Trans. Multimedia*, vol. 8, pp. 390–404, Apr. 2006.
[20] C. Chan, S. Wee, and J. Apostolopoulos, “On optimal embedded schedules of JPEG-2000 packets,” in *Proc. IEEE Int. Conf on Image Processing (ICIP ’06)*, (Atlanta, GA, USA), pp. 785–788, Oct. 2006.
[21] S. Wu, S. Dumitrescu, and N. Zhang, “On multirate optimality of JPEG2000 code stream,” *IEEE Trans. Image Process.*, vol. 14, pp. 2012–2023, Dec. 2005.
[22] Y. Wang, M. van der Schaar, S.-F. Chang, and A. C. Loui, “Classification-based multidimensional adaptation prediction for scalable video coding using subjective quality evaluation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 1270–1279, Oct. 2005.
[23] C. Chan, S. Wee, and J. Apostolopoulos, “Multiple distortion measures for scalable streaming with JPEG2000,” in *Proc. IEEE Int. Conf on Acoustics, Speech, and Signal Processing (ICASSP ’07)*, (Hawaii, USA), pp. 705–708, Apr. 2007.
[24] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer-Verlag, 2004.
[25] J. Apostolopoulos, “Secure media streaming & secure adaptation for non-scalable video,” in *Proc. IEEE Int. Conf on Image Processing (ICIP ’04)*, vol. 3, (Singapore), pp. 1763–1766, Oct. 2004.



Carri W. Chan Carri W. Chan is currently a Ph.D. candidate in the Electrical Engineering Department at Stanford University. She received her B.S. degree in Electrical Engineering from the Massachusetts Institute of Technology in 2004 and her M.S. degree in Electrical Engineering from Stanford University in 2006. Her research interests include stochastic modeling and optimization of communication networks. Her recent focus has been on scheduling problems for packetized multimedia streaming over wireless networks.

Susie Wee Susie Wee is currently the Vice President of the HP Experience Software Business. She has been with HP for over 11 years, previously as the Director of the Mobile and Media Systems Lab in HP Labs. She has worked closely with many HP businesses and she has led international research collaborations with NTT DoCoMo in Japan, Infocomm Institute of Research in Singapore, and universities such as MIT. Susie’s research interests are in video processing, image security and multimedia streaming. Computerworld named her as one of the ”Top Innovators” under age 40 in 2007 and she was named one of the ”Top Innovators” under age 35 by MIT Technology Review in 2003. Susie was recently awarded the Technical Excellence Award from INCITS for her work on creating the international standard on JPEG2000 Image Security (JPSEC). She holds over 25 granted patents and was a consulting assistant professor at Stanford University co-teaching a graduate class on digital video processing. Susie was formerly an associate editor of the IEEE Transactions on Image Processing and the IEEE Transactions on Circuits, Systems, and Video Technology. Susie received her Bachelor of Science, Master of Science and Ph.D. degrees in Electrical Engineering and Computer Science from MIT.

John Apostolopoulos John G. Apostolopoulos (S’91, M’97, SM’06, F’08) received the B.S., M.S., and Ph.D. degrees in EECS from MIT. He joined Hewlett-Packard Laboratories in 1997, where he is currently a Distinguished Technologist and Lab Director for the Multimedia Communications and Networking Lab. He also teaches and conducts joint research at Stanford University, where he is a Consulting Associate Professor of EE, and also regularly lectures at MIT. In graduate school, he worked on the U.S. Digital TV standard and received an Emmy Award Certificate for his contributions. He received a best student paper award for part of his Ph.D. thesis, the Young Investigator Award (best paper award) at VCIP 2001 for his work on multiple description video coding and path diversity, was named ”one of the world’s top 100 young (under 35) innovators in science and technology” (TR100) by Technology Review in 2003, and was co-author for the best paper award at ICME 2006 on authentication for streaming media. His work on media transcoding in the middle of a network while preserving end-to-end security (secure transcoding) was recently adopted by the JPEG-2000 Security (JPSEC) standard. He currently serves as chair of the IEEE IMDSP and member of MMSP technical committees, and recently was general co-chair of VCIP’06 and technical co-chair for ICIP’07. His research interests include improving the reliability, fidelity, scalability, and security of media communication over wired and wireless packet networks.