Adaptive Streaming of Layer-Encoded Multimedia Presentations

David A. Turner
Dapartment of Computer Science
California State University
San Bernardino, CA
dturner@csusb.edu

Keith W. Ross
Institut Eurécom
Sophia Antipolis, France
ross@eurecom.fr

Abstract

In the context of a communication network with no QoS guarantees, we describe the problem of adaptive streaming of layer encoded multimedia presentation data as a two-phase decision problem. In phase one the application transfers only base layer data that comprise a presentation of minimum quality, which is stored at the client. When the application determines that commencing play out will result in an uninterrupted presentation of at least minimum quality, it does so, and then transitions into phase two. The application then loops on the decision on which data to send next: another base layer, or an enhancement layer. It does this by considering the deliverability and play-out quality within a sliding window of media units. We present two different algorithms for making this decision, based on two different presentation quality metrics: the *total quality* metric, which yields an optimization problem that can be solved with dynamic programming, and the *refined max-min* metric, which yields a computationally inexpensive algorithm for computing an optimal decision. We also consider the problem of progressively rendering static objects after their start times as a means of improving presentation quality. We compare the various approaches with a slide show presentation with a randomly generated sequence of layer-encoded JPEG images, and an Ogg Vorbis audio stream.

1 Introduction

A multimedia presentation consists of a collection of objects, with each object having one or more rendering intervals within the presentation timeline. These intervals specify the objects' start times and end times relative to the presentation timeline. For example, the presentation might consist of an audio stream that is played continuously while a sequence of images is displayed. Other examples might include one or more video clips (played simultaneously or sequentially), a stream of music, animation, etc.

In this paper we consider the problem of streaming a layer-encoded multimedia presentation from a server to an arbitrary client over a communication network with no QoS guarantees, such as the global IP network. Studies of TCP bandwidth on the Internet [9] demonstrate the non-stationarity of TCP throughput. For this reason, the application must continuously monitor packet loss and delay characteristics, and use this in adjusting its expectation of future bandwidth. We assume the application has such an estimator, which will necessarily be related to the transport protocol used, whether it be TCP or an application specific protocol implemented over UDP.

The advantage of layer-encoding multimedia presentation data is that some layers can be dropped to reduce the size of the data representation, which will increase the speed at which the presentation can be transported over the network. With faster transmission, the end user is saved from waiting for presentation data to be pre-fetched into a playback buffer. However, fewer layers means lower quality, so we would like to send as many layers as possible while keeping the start up latency to a minimum.

We assume that a presentation comprised of only the first layer of each object represents a presentation of minimum acceptable quality. We will use the term *base layer* when referring to the first layer, and *enhancement layer* when referring to higher layers, which contribute to quality but are not required for achieving the minimum quality level.

Our first priority is to minimize the start up delay for play out of a presentation comprised only of base layers. After establishing the minimum start up delay, our second priority is to improve the quality of the presentation by sending enhancement layers. Corresponding to these two priorities, we have two decision phases through which the application passes. In the first phase, the application simply sends base layers for objects in the order of their first appearance in the presentation. While it is sending base layer data, it collects information about the available bandwidth and uses it to decide when to start playing the presentation. After the presentation starts playing, the application enters into the second phase in which it loops on the decision of which layer to send next.

We allow each layer level of each object to have a general quality value. In this manner, there is considerable flexibility in defining the quality of a rendered object. For example, the quality of an object could be the percentage of layers rendered, the percentage of compressed bits rendered, or the mean squared error of the rendered object.

There are several natural measures for the overall quality of a presentation. One natural measure is the sum of the qualities of the individual objects, which we refer to as the total quality of the presentation. For this total criterion, we develop a dynamic programming algorithm that computes the number of layers to send of each object to maximize the presentation quality, constrained by playback deadlines and a lower bound on expected future bandwidth. However, in its attempt to generate a presentation with highest total quality score, the algorithm may produce presentations with highly varying object qualities. To avoid this result, we introduce a new criterion that we refer to as the refined max-min criterion. This criterion is known as lexigographic optimality in the operations research literature [18], and as max-min fairness in the literature on Available Bit Rate (ABR) networks. The refined max-min criterion strives to equalize the quality values of all objects while improving quality in a uniform manner when extra bandwidth remains available. We also consider the problem of gradually rendering layers between their start and end times. Gradual rendering provides more flexibility at the cost of missing some start-time deadlines.

In the context of stored VBR-encoded video, several papers have studied the transmission of video from a server to a client with finite storage; a partial list includes [7, 10, 11, 12, 13, 14, 15, 16]. All of these papers assume that the video is encoded with one layer. These papers have examined a variety of smoothing and pre-fetching schemes that minimize bandwidth usage for fixed start-up delay and finite client storage capacity. Although finite client storage remains an important issue for mobile handsets, the large majority of Internet users today have abundant local storage. The schemes that we have developed in this paper target these users.

Recently, Zhao, Willebeek-LeMair and Tiwari have considered "malleable multimedia presentations" [6, 7]. Their model is similar to ours in that the objects are assumed to be encoded in layers. A central assumption in their work is that client storage is finite. They investigate the tradeoff between the size of client storage and the number of layers that can be transmitted, assuming that the transmission bandwidth is limited. In [7], the authors assume all objects have the same number layers and only consider policies that transmit the same number of layers for each object.

They propose a binary search algorithm to search through the number of layers in order to find the maximum number of layers that can be sent while satisfying the bandwidth constraint. In [6] they provide an enhancement algorithm that adds one layer to a subset of the objects while remaining feasible.

Our work differs from [6] and [7] in many respects. First, we place the problem within the context of a two-phase decision problem in which start up delay is minimized, which is not done in [6] and [7]. Second, we allow for general quality values for each layer for each object. References [6] and [7] only consider the special case where quality equals number of rendered layers, which is in many circumstances not an accurate measure of the quality. Third, we examine optimization criteria that are different from simply finding the maximum number of layers that can be transmitted for all objects. We consider the natural total quality criterion and show how optimizing total quality can be formulated as a dynamic programming problem. We also propose a refined max-min criterion, which makes efficient use of the available bandwidth while striving to maximize the worst-case quality for rendered objects. We present numerical testing with progressive JPEG data to investigate and compare the different quality value definitions and optimization criteria. We examine the use of the refined max-min criterion for Ogg Vorbis video streams, which is a non-proprietry high-fidelity audio encoding, and we describe how the streaming methodolgy works for video streams. Finally, we introduce progressive rendering of static objects into the optimization objective, and study how progressive rendering influences the optimal policy.

2 Determining Startup Delay

Frequently a distinction is made between continuous and discrete media; audio and video are classified as continuous, while still images and text are classified as discrete. For our purposes, the main difference is that the rendering of a discrete object can be started only after the data comprising the object arrives at the client in its entirety, but the rendering of a continuous media object can be started before all of its data arrives at the client. In our approach, however, we decompose continuous media into separate discrete objects with relatively short rendering periods. This allows us to treat all data components of the presentation as atomic units with arrival deadlines at the client equal to the starting point of their rendering intervals.

In the case of still images, the entire image becomes an object in our representation, and this object has a play out deadline at the client. In the case of audio, compressed audio streams must be organized into a sequence of packets that represent short contiguous play out interevals, which is generally the case with audio compression schemes. In the case of video, the stream needs to be organized into objects that are independently decodable. In the case of MPEG video, P and B frames can not be treated seperately from the I frames that precede them. Thus, for the purposes of this paper, the smallest possible organizational unit of an MPEG II video stream is an I frame followed by all of its successive B and P frames, up to, but not including, the next I frame.

Because we are transmitting data over a network with no QoS guarantees, after a communication channel is established, the application does not know the rate at which it can transmit data from source to destination. For this reason, the application begins by transmitting only base layer data, which it stores in a pre-fetch buffer at the client until it determines that it can safely begin play out of the presentation without the threat of the presentation stalling.

While the application is transmitting base layers, it performs two other tasks in parallel. First, it records a packet transmission history, which it uses for predicting future bandwidth. Second, it loops on the decision whether to begin playing the presentation.

We do not assume any particular transport protocol, such as TCP or a particular UDP-based scheme. However, for whatever transport mechanism is employed, we assume the application has a reasonable method of estimating from the packet transmission history a lower bound for expected future bandwidth. We let $B(a, b)$ represent a lower bound for the number of bits the application expects can be delivered from server to client in the time interval $[a, b]$.

i	index used to label the presentation objects
N	the number of objects in the presentation
L_i	the total number of layers for object i
j_i	layers of object i to be transmitted (and rendered)
$B(a, b)$	estimated lower bound on bandwidth on $[a, b]$
t_i	deadline for the arrival of layers of object i

Figure 1: Notation introduced in Sec. 2

Let N represent the number of discrete objects in the presentation after decomposition of the continuous media, and let t_i represent the starting point of the initial rendering interval of object

i. We assume the objects are ordered by increasing values of their deadlines. We measure time relative to the start of the presentation, so that the presentation starts at time $t = 0$. We let L_i represent the number of layers in object i.

Suppose the application has delivered the base layers of objects 1 through $q - 1$ to the client, and is now considering whether to start play out. Let $x_{i,j}$ be the number of bits in the j^{th} layer of object i. If the application starts play out now, all of the unsent base layers will arrive on-time if the cumulative bits needed at each deadline is less than or equal to the cumulative bits that can be delivered. Thus, the application starts play out if the following inequalities are satisfied:

$$\begin{aligned}
&x_{q,1} \leq B(0, t_q) \\
&x_{q,1} + x_{q+1,1} \leq B(0, t_{q+1}) \\
&\vdots \\
&x_{q,1} + \ldots + x_{N,1} \leq B(0, t_N)
\end{aligned}$$

If this system of inequalities is not satisfied, the application does not begin play out, but continues to pre-fetch additional base layers.

3 Layer Selection

After the application gives the command to start playback, it transitions into phase two, in which it loops on the decision of which layer to send next. To make this decision, the application continues to record transmission statistics and refines its estimate of a lower bound on future bandwidth. With this estimate, it determines a sequence of layers P (which we call a transmission policy) that can be delivered on-time (which we call feasibility) and that maximizes an objective measure of presentation quality $Q(P)$. The first layer in the sequence comprising this policy is chosen to be transmitted next.

Suppose that the rendering of the first r objects have already started, so the application only needs to concern itself with scheduling the delivery of layers from objects $r + 1$ to N. To simplify notation, we re-label these objects as 1 through M.

A transmission policy specifies the number of layers to send for each object, which we represent as an M-dimensional vector P whose i^{th} component j_i represents the number of layers of object i to send to the client. We call a policy *feasible* if all of the bits it sends arrive at the client prior to their deadlines.

M	the number of non-rendered objects
P	the transmission policy $(j_1, \ldots, j_M)$
$Q(P)$	presentation quality resulting from P
b_i	expected bandwidth between deadlines t_{i-1} and t_i
$y_i(j)$	unsent bits of object i in layers 1 through j
$q_i(j)$	quality of object i when layers 1 through j are rendered

Figure 2: Notation introduced in Sec. 3

Let $b_i = B(t_{i-1}, t_i)$, the application's estimate of the number of bits that can be transmitted between the deadlines t_{i-1} and t_i. At each deadline, the cumulative number of bits needed must be less than or equal to the cumulative bits that will be transmitted. Thus, if we let $y_i(j)$ equal the unsent bits of object i that appear in layers 1 through j, then policy P is feasible if the following system of M inequalities hold.

$$\begin{aligned}
&y_1(j_1) \leq b_1 \\
&y_1(j_1) + y_2(j_2) \leq b_1 + b_2 \\
&\vdots \\
&y_1(j_1) + \ldots y_M(j_M) \leq b_1 + \ldots + b_M
\end{aligned}$$

Now that we have defined the set of feasible policies, we need to determine which of these policies are optimal, that is, which policies result in presentations with the best quality. In the following two sections, we consider two measures of overall presentation quality, and provide algorithms that converge to optimal policies under these measures. In both sections, we assume a general measure $q_i(j)$ for the quality of an individual object i when layers 1 through j are used for its rendering.

4 The Total Quality Criterion

After the application begins play back, and enters into phase 2, it loops on the decision regarding which layer to send next. To make this decision, it uses its estimate of future bandwidth to compute an optimal transmission policy, that is, a feasible transmission policy that maximizes the overall quality of the presentation. In this section, we use dynamic programming to develop an efficient algorithm for the determination of the optimal transmission policy with respect to the *total quality* metric, which equates the overall quality of a presentation with the sum of the qualities of its component objects. That is,

$$Q(P) = \sum_{i=0}^{M} q_i(j_i)$$

A brute-force determination of $Q(P)$ requires $L_1 L_2 \ldots L_M$ calculations, which is not polynomial-bounded in the number of objects. However, there exists a recursive formulation of the problem that yields an algorithm that terminates in polynomial time. To do this, we first develop some terminology, which we use to state a theorem. Then we show how the theorem provides an efficient method of solving the optimization problem. A proof of the theoreom is provided in the appendix.

Suppose we only want to send objects i through M, and that we have s surplus bits of bandwidth available to do this, in addition to the $b_i, \ldots, b_M$ bits that are available in the intervals terminating at deadlines $t_i, \ldots, t_M$. The sub-policy $(j_i, \ldots, j_M)$ is feasible if it satisfies the following deadline constraints:

$$\begin{aligned} & y_i(j_i) \leq s + b_i \\ & y_i(j_i) + y_{i+1}(j_{i+1}) \leq s + b_i + b_{i+1} \\ & \vdots \\ & y_i(j_i) + \ldots + y_M(j_M) \leq s + b_i + \ldots + b_M \end{aligned} \tag{1}$$

Let $\Omega_i(s)$ be the set of all such feasible sub-policies, that is, all sub-policies that satisfy (1). We define $f(i, s)$ to be the maximum total quality attainable over the set of feasible sub-policies. That is,

$$f(i, s) = \max_{\Omega_i(s)} \Big(q_i(j_i) + \ldots + q_M(j_M) \Big). \tag{2}$$

By definition, $f(1, 0)$ is the maximum quality for the presentation. We develop a recursive expression for f that will allow us to compute $f(M, s)$, $f(M-1, s)$, $\ldots$, $f(2, s)$, $f(1, 0)$, in that order. For this purpose, define $\mathcal{X}_i(s)$ to be the set of all possible values for the number of layers of object i that can be sent within a bandwidth of s (surplus) bits plus the bits that can be transmitted between deadlines t_{i-1} and t_i. That is,

$$\mathcal{X}_i(s) = \Big\{ j \mid y_i(j) \leq s + b_i \Big\}. \tag{3}$$

Theorem 1 *$f(i, s)$ can be expressed recursively as follows:*

$$f(M, s) = \max_{j \in \mathcal{X}_M(s)} \Big(q_M(j) \Big), \tag{4}$$

and for $i = M-1, \ldots, 1$,

$$f(i, s) = \max_{j \in \mathcal{X}_i(s)} \Big(q_i(j) + f\Big(i+1, s + b_i - y_i(j)\Big) \Big). \tag{5}$$

This theorem provides a way to compute an optimal policy using the following procedure:

1. Use (4) to compute and store $f(M, s)$ for values of s that range from 0 to the total number of presentation bits (incrementing s by some reasonable value, such as 100).

2. Similarly, for varying values of s, use (5) to compute and store $f(M-1, s)$, $f(M-2, s)$,..., $f(2, s)$, in that order.

3. Compute $\widehat{j}_1$ that maximizes (5) for $f(1, 0)$, that is, $\widehat{j}_1$ maximizes $q_1(j) + f(2, b_1 - y_1(j))$ over $\mathcal{X}_1(0)$. Set $\widehat{s}_1$ to the unused bandwidth to be carried into the next interval, that is, $\widehat{s}_1 = b_1 - y_1(\widehat{j}_1)$.

4. Repeat the following for $i = 2, \ldots, M-1$:

   - Compute the value $\widehat{j}_i$ that maximizes $q_i(j) + f(i+1, \widehat{s}_{i-1} + b_i - y_i(j))$ over $\mathcal{X}_i(\widehat{s}_{i-1})$.

   - Let $\widehat{s}_i$ represent the unused bandwidth to carry into the next interval, that is, $\widehat{s}_i = \widehat{s}_{i-1} + b_i - y_i(\widehat{j}_i)$.

5. Compute the value $\widehat{j}_M$ that maximizes $q_M(j)$ over $\mathcal{X}_M(\widehat{s}_{M-1})$.

The optimal transmission policy will be $(\widehat{j}_1, \ldots, \widehat{j}_M)$. This algorithm has complexity $O(L \cdot M \cdot S)$, where L is an upper bound for the number of layers in each object, M is the number of objects, and S is the number of presentation bits.

We will use the algorithm developed in this section to compute, in Sec. 6, optimal transmission policies for two different object quality functions at varying levels of bandwidth. We will compare these policies with those generated from a different algorithm that we develop in the next section based on a different presentation quality metric, called the max-min quality metric. Both of these algorithms generate optimal policies under their respective criteria, and both can be used for scheduling the delivery of presentation layers after play out has begun.

5 The Refined Max-Min Criterion

One possible measure of overall presentation quality is simply to take the worst object quality. A feasible policy would then be optimal if it maximizes the minimum quality across all objects in the presentation. We call this the *max-min criterion.*

Although the max-min criterion is a natural choice for a criterion of optimality, it typically provides policies that do not allocate all the available bandwidth. Consequently, sending additional layers for some objects may be possible, which, although not increasing the minimum quality, clearly improves the overall quality of the presentation. Also, in general, less cumulative bandwidth will be available for transmitting objects with earlier deadlines, thus the minimum attainable quality will be dominated by bandwidth available for the early objects. We now consider a *refined max-min criterion* that overcomes these inadequacies.

With the refined max-min criterion, we represent the overall quality of the delivered presentation by a vector of object qualities, appearing in increasing levels of quality. For example, suppose we have a presentation with three objects, a transmission policy of $P = (2, 4, 3)$, and quality values $q_1(2) = 16.3$, $q_2(4) = 30.4$, and $q_3(3) = 20.5$, so that $Q(P) = (16.3, 20.5, 30.4)$. We let a_i represent the object that takes the i^{th} position in the quality vector, so that in our numerical example we have $a_1 = 1$, $a_2 = 3$, and $a_3 = 2$. In general, we can express the overall presentation quality as

$$Q(P) = (q_{a_1}(j_{a_1}), \ldots, q_{a_M}(j_{a_M})).$$

It is important to note that if a policy is optimal for the refined max-min criterion, then it is also optimal for the max-min criterion. However, the converse is not generally true. Thus, the refined max-min criterion is a more sensible measure for the overall quality of a presentation, because in addition to satisfying the max-min criterion, it better exploits the available bandwidth to improve the quality of the presentation.

We now present an algorithm that determines the optimal policy under the assumption that the quality values $q_i(j)$ are distinct for all values of i and j. Quality measures that utilize the length of time rendered, root mean square, or number of bits, typically fulfill this assumption. The algorithm (Fig. 3) initializes a policy vector $(j_1, \ldots, j_M)$ to the layers that have already been sent to the client. Then it sets to 1 the layer counters j_i of all objects that have not yet had their base layer sent. This will result in a feasible policy, because the application runs this algorithm only when there is adequate bandwidth to deliver all base layers prior to their deadlines. Then, the algorithm enters a loop in which it tries to add a layer to the object with lowest quality. If adding a layer to this object results in a feasible policy, then the layer counter for this object is incremented. On the other hand, if adding a layer results in a non-feasible policy, then its layers are held fixed, and no longer considered (by removal from S) for a possible quality improvement. A proof that Fig. 3 produces an optimal transmission policy is provided in the appendix.

```
do for $i = 1, \ldots, M$
    $j_i$ = number of layers sent of object $i$
    if $j_i == 0$ then $j_i = 1$
$S = \{1, \ldots, M\}$
do while $S$ is not empty
    find $k \in S$ s.t. $q_k(j_k) \leq q_i(j_i)$ for all $i$ in $S$
    if $(j_1, \ldots, j_k + 1, \ldots, j_M)$ is feasible then
        $j_k = j_k + 1$
    else
        remove $k$ from $S$
```

Figure 3: Refined max-min algorithm

For object quality measures that map into a relatively small range, such as $q_i(j) = j$, the refined max-min algorithm may not converge to an optimal policy, because it does not properly resolve quality ties. We propose (and evaluate in Sec. 6) the following tie-breaking heuristic: in the presence of a quality tie, choose the object whose next layer has the smallest number of bits. Intuitively, this heuristic makes sense, because we are improving the quality by one layer with the least expenditure of bandwidth.

In the next section, we use a sample presentation to compute the optimal policies under the max-min criterion for two different object quality functions, and for varying levels of bandwidth. We also do this for the total quality criterion discussed in Sec. 4, and use the results to compare the two approaches. We also examine refined max-min optimal transmission polices for a video stream encoded using the Ogg Vorbis audio compression scheme. Then in Sec. 7 we look at a method to extend the refined max-min approach to better utilize available bandwidth by permitting progressive rendering of static objects.

6 Experimental Results

In this section we report some of our experimental results regarding the difference between the policies generated by our total quality and refined max-min algorithms. We will see how well the refined max-min algorithm performs under the tie-breaking heuristic, and we will compare the

Table 1: Image data for slide show presentation

image	deadline	bytes by layer number										total
	(secs)	1	2	3	4	5	6	7	8	9	10	
1	0	1321	1956	457	306	1926	2899	246	492	415	5210	15228
2	18	2966	8026	7479	14341	566	25803					59181
3	36	11118	14782	1594	2635	14471	28445	2556	3060	3830	73974	156465
4	51	11223	22622	12931	35286	2106	66897					151065
5	62	10536	17380	1007	1600	14285	30474	2550	3521	4839	81178	167370
6	76	3473	3666	602	569	1700	6880	826	978	922	19360	38976
7	95	4596	7312	852	629	2375	7988	952	1616	1645	14893	42858
8	107	9253	7322	818	1200	3068	16640	2587	2025	2482	68278	113673
9	124	4424	4969	1554	1153	2284	6597	829	1613	1473	14705	39601
10	133	13221	28030	2438	4026	45614	58405	2621	5035	8037	116863	284290

policies generated by the two algorithms using different individual object quality functions. We are not reporting on the two-phase decision problem.

We assembled a slide show presentation with 2 black and white and 8 color JPEG images in order to compare the refined max-min criterion with the total quality criterion under two different object quality measures. We encoded the images using the Independent JPEG Group's library [17], which by default encodes color images into 10 layers and black and white images into 6 layers. The two object quality measures were a layer-oriented measure, which equates quality with the ratio of layers rendered to the total number of layers in the object, and a bit-oriented quality measure, which equates quality with the ratio of bits rendered to total bits from all layers. Note that under the bit-oriented measure, convergence to an optimal refined max-min policy is guaranteed. The two algorithms are independent of the object quality metric. The bit-oriented and layer-oriented quality metrics are used to illustrate the essential differences between the two criteria for optimality. Other quality metrics, such as the mean squared error, and perceptual-based metrics will undoubtedly produce similar differences. We chose rendering times that reflect a quick-paced slide-show presentation. Table 6 shows the number of bytes in each layer of each image, and the deadlines for the arrival of each image relative to the presentation timeline.

The first object quality measure is the ratio of layers rendered to the total number of layers in the object, i.e., $q_i = j/L_i$. We call this the *layer-oriented quality measure*. We use this rather than the number of layers, because the objects in the presentation are not encoded into the same number of layers. The layer-oriented quality measure generally results in many ties, because many objects in the presentation will be encoded with the same number of layers. Thus, the policy that the refined max-min algorithm produces may be sub-optimal.

The second object quality measure is the ratio of bits rendered to total bits across all layers. We call this the bit-oriented quality measure. For the bit-oriented quality measure, we have:

$$q_i(j) = \frac{\sum_{k=1}^{j} x_{ij}}{x_i} \text{ , where } x_i = \sum_{j=1}^{L_i} x_{ij}$$

This is a somewhat natural measure in that the number of encoded bits, in a loose sense, represents the "information" in the layers; we are therefore associating quality with rendered information.

We suppose that there is approximately a constant 24 Kbps of bandwidth available for the presentation, and we fix the start up delay at 5 seconds. Fig. 4 shows the percentage of layers sent for each of the ten objects in the presentation for the refined max-min and total quality criteria under the layer-oriented quality measure. The resulting policies of the two algorithms appear to agree in general regarding which images should be weak (in terms of percentage of layers rendered) and which should be strong. However, the refined max-min algorithm produces a presentation with more uniform image qualities.

Figure 4: Percentage of layers sent by criterion under the layer-oriented quality measure

We can also see from Fig. 4 that the worst case object quality is 50 percent for the refined max-min criterion. The ordinary max-min criterion would have stopped at this point, generating a policy that transmits 50% of the layers for each object. The refined max-min criterion enables the presentation to display more layers than an ordinary max-min criterion while still respecting the max-min philosophy.

Fig. 5 shows the percentage of bits sent for each object under using the bit-oriented quality

Figure 5: Percentage of layers sent by criterion under the bit-oriented quality measure

measure. Here the two algorithms give strikingly different policies. While the refined max-min algorithm continues to distribute relatively equal importance across all objects, the total quality algorithm selects a highly non-uniform distribution. The worst case object quality for the refined max-min criterion is approximately 30 percent of the object's bits, while that of the total quality criterion is less than 10 percent.

In order to further examine the differences between the two criteria and the two quality measures, we computed the optimal policies for the sample presentation while varying the level of bandwidth. We plotted three different summary statistics related to presentation quality: minimum percentage of layers rendered by bandwidth (Fig. 6), bandwidth utilization (Fig. 7), and average percentage of layers rendered by bandwidth (Fig. 8). Note that the legend appearing in Fig. 6 applies to all three figures.

Fig. 6 demonstrates that the refined max-min criterion is superior to the total quality criterion with respect to minimizing the worst quality, which isn't surprising, because this objective is its primary motivation. It should also be noted that the layer-oriented quality measure performs better than the bit-oriented quality measure. There are a few points where the total quality criterion with the layer-oriented measure performs better than the refined max-min criterion with the bit-oriented measure, but in general the refined max-min criterion performs better with both quality measures.

After maximizing the minimum object quality, our second motivation was to improve the presentation quality by using as much of the additional bandwidth as possible, while trying to minimize the worst case of those images that could be helped. Fig. 7 shows the bandwidth utilization (band-

Figure 6: Minimum percentage of layers rendered by bandwidth

width consumed ÷ available bandwidth) for the two criteria with the two quality measures. Here, all methods show increased variance in the upper bandwidth region, which is explained by the large amount of data concentrated in the final layer of each image. The plot shows that the refined max-min criterion works slightly better with the bit-oriented measure.

Fig. 8 shows the average percentage of layers rendered by bandwidth levels. Here the total quality criterion based on the layer-oriented quality measure is superior to the other methods, especially for the low- and mid-range bandwidths. The total quality criterion with the bit-oriented quality measure also appears to do well in the lower bandwidths, but gives weaker results for higher levels of bandwidth. At high levels of bandwidth, the various methods converge, but the refined max-min criterion based on the layer-oriented quality measure converges the most quickly. The refined max-min criterion with the bit-oriented quality measure is the clear loser in this comparison.

Clearly, the Total Quality, Max-min, and Refined Max-min criteria can produce rather different optimal policies. We believe the refined max-min criterion is superior to both the total quality and max-min criteria. Nevertheless, in order to make a more definite conclusion, subjective testing with human subjects is needed with other quality metrics, such as the mean-squared error.

Figure 7: Bandwidth utilization by criterion/quality measure

Figure 8: Average percentage of layers rendered by bandwidth

To get a clearer idea of the type of transmission policies the Refined Max-min criterion produces, we examined an Ogg Vorbis audio stream of music. Ogg Vorbis [21] is an open, non-proprietary compressed audio format for mid to high quality audio and music at fixed and variable bitrates from 16 to 128 kbps/channel, and is therefore roughly similar in abilities to MPEG-4 (AAC), MPEG-4 audio (TwinVQ), WMA and PAC. We examined a two-minute music stream comprised of 96 Ogg packets, or objects in our nomenclature. Fig. 9 shows the number of bits contained in the objects comprising this stream. These Ogg packets do not represent fixed-length playout intervals, but range from 0.5 to 3.5 seconds in duration.

Figure 9: Media units of an Ogg Vorbis music stream

The 96 component objects are each structured into 10 layers. In the case that the client pre-fetch buffer has a capacity of 1000 KB, the refined max-min optimal transmission policy forms a non-decreasing sequence of object quality scores, as shown in Fig. 10. The one exception is the first object, which contains non-scalable information requried by the decoder, and thus appears with 100% resolution.

Figure 10: Object quality scores for 1000 KB pre-fetch buffer

It is demonstrated in [3] that optimal refined max-min transmission policies for continuously scalable media streams will always form a non-decreasing sequence of quality scores. In our example, the use of 10 discrete layers is sufficient to produce a result similar to the continuously scalable case.

When the client pre-fetch buffer is constrained to 300 KB, the resulting optimal transmission policy is no longer nondecreasing, as Fig. 11 illustrates. In this case, the streaming server must decrease the quality of objects in order to avoid buffer overflow. This problem is fully described out in [3] and [4].

Figure 11: Object quality scores for 300 KB pre-fetch buffer

7 Progressive Rendering

Now we consider the benefits and methodology of progressive rendering of static object data, such as JPEG images. In progressive rendering, the object layers are permitted to arrive after the beginning of the object's rendering period, but before the end of the rendering period. In this way, the client can improve the object's quality by rendering additional layers that arrive late.

To see that a transmission policy that includes progressively rendered objects has value, consider a slide show presentation of one minute duration that includes three images. The first image is displayed immediately and is rendered for the entire 60-second length of the presentation. The second image is rendered 20 seconds later and is rendered for the remaining 40 seconds. The third image is displayed 40 seconds into the presentation and is rendered for the remaining 20 seconds. Suppose the images are encoded into 6 layers.

Because the first image is to be displayed at the start of the presentation, the length of time needed for the transport of its base layer will determine the presentation start up delay. If the images are of equal size, and the bandwidth in 10 seconds is adequate to send all 6 layers of an image, then play out of the presentation will be $(1, 6, 6)$, that is, 1 layer of image 1, 6 layers of image 2, and 6 layers of image 3.

However, if we allowed an image to be progressively rendered in increasing degrees of quality, we could play out the presentation $(6, 6, 6)$. In this scenario, image 1 is rendered with 1 layer at time $t = 0$, and then is rendered with increasing quality as additional layers arrive during the next 10 seconds. At time $t = 10$ the transmission of layers of image 2 begins. All layers of images 2 and 3 arrive before the starting points of their rendering intervals, and thus are rendered with full quality. Clearly, this presentation has better overall quality than the one without progressive rendering. In general, presentations with a static object whose rendering begins at the beginning of the presentation will only have its base layer rendered.

Besides rendering more layers of static objects near the beginning of the presentation, another benefit of progressive rendering is to increase bandwidth utilization near the end of the presentation. After the last object is delivered, it is possible to improve the quality of static objects that remain displayed by sending additional enhancement layers.

To see that progressive rendering is applicable for objects other than the first and the last, consider a presentation with three objects, each having 10 layers with 10K bits per layer. The presentation starts with a period of silence in which 30K of bandwidth is available. Then, object 1 is rendered for a length of time in which 30.01K of bandwidth is available. At the end of this period, the rendering of object 1 ceases and the rendering of object 2 begins. Then, there is 200K of bandwidth available during the rendering period of object 2. When the rendering period of object 2 ends, the rendering period of object 3 begins. (In our example, it is not necessary to specify the length of the rendering period of object three.) Without progressive rendering, the optimal refined max-min policy will be to send 3 layers of the first object (consuming the 30K of bandwidth in the first interval), 3 layers of the second object (consuming the 30K of bandwidth in the second interval), and 10 layers of the third object (consuming 100K of bandwidth in the third interval). In summary, the optimal refined max-min policy without progressive rendering will be (3, 3, 10).

On the other hand, if we permit progressive rendering, it is possible to render the policy (6, 10, 10). With this policy, the server transmits 26 layers back-to-back with the available bandwidth. First, it transmits the six layers of object 1, then the 10 layers of object 2, and finally the 10 layers of object 3. The first 3 layers of the first object are rendered at the beginning of its rendering interval, followed by 3 additional layers that are progressively rendered throughout the interval. When the rendering interval of the second object begins, no layers will be available, but each of the object's 10 layers will be transmitted and progressively rendered during the initial one third of the interval. The remaining bandwidth will then be used to transmit 10 layers of the third object, which will be rendered on time.

One drawback of policy (6, 10, 10) over (3, 3, 10) is that now the first three layers of the second object will arrive late relative to the start of their rendering interval. Thus it could be argued that this delay makes the progressive rendering policy inferior. In this case, one could use the policy (3, 10, 10), in which the first 3 layers of the second object are available at the start of its rendering interval, as in policy (3, 3, 10), and the 7 additional layers are progressively rendered.

However, a more serious objection to policy (6, 10, 10) is that layer 6 of the first object is used for an insignificant amount of time, but results in a significant delay in the rendering of the second object. Intuitively, policy (5, 10, 10) is better than (6, 10, 10), because it avoids sending a layer which is rendered for an insignificant amount of time and reduces the rendering delay for the layers of the second object. The algorithm we present does not recognize this trade-off, and converges to policy (6, 10, 10). However, if we modify the definition of feasibility so that it rejects layers that arrive excessively late — such as 1 second prior to the end of their rendering interval — then the algorithm will converge to policy (5, 10, 10) in this example.

For progressive rendering, we now consider a policy P to be feasible if the bits sent arrive prior to the end, rather than the start, of their rendering intervals. For illustrative purposes, we consider the following natural definition of quality for progressive rendering:

$$q_i(j_1, \ldots, j_i) = \frac{1}{L_i} \sum_{j=1}^{L_i} \frac{v_{ij}}{w_i}, \text{ where}$$

v_{ij} is the rendering time of layer j of object i, and w_i is the length of the rendering interval for object i. The calculation of v_{ij} depends on the number of layers chosen for object i and the objects preceding object i, which is why we express q_i as a function of these layers.

The refined max-min algorithm *without* progressive rendering converges, because adding a layer to the object with the worst quality will not degrade the quality of the other objects. But this may not be the case with our new definitions of feasibility and quality, because at some point within the execution of the algorithm, it may be possible to add a layer to the object with minimum quality that results in the delayed arrival of succeeding layers, thus degrading their qualities. We propose the heuristic of testing for degradation in overall quality before adding a layer. The algorithm for progressive rendering with our proposed heuristic is shown in Fig. 12.

```
do for $i = 1, \ldots, M$
    $j_i$ = number of layers sent of object $i$
    if $j_i == 0$ then $j_i = 1$
$S = \{1, \ldots, M\}$
do while $S$ is not empty
    find $k \in S$ s.t. $q_k(j_k) \leq q_i(j_i)$ for all $i$ in $S$
    $P = (j_1, \ldots, j_M)$
    $P' = (j_1, \ldots, j_k + 1, \ldots, j_M)$
    if $P'$ is feasible AND $Q(P') > Q(P)$ then
        $j_k = j_k + 1$
    else
        remove $k$ from $S$
```

Figure 12: Refined max-min algorithm with progressive rendering

We used the same presentation data in Table 6 in Sec. 6 to compare the layer-oriented refined max-min algorithm with and without progressive rendering. Fig. 13 shows the percentage of layers rendered for each object under the two methods. Note that the policy with progressive rendering is better than the policy without progressive rendering in the sense of the refined max-min criterion. Interestingly, the progressive rendering algorithm decreases the number of layers in images 1 and 2 in order to send more layers of image 5. It also decreases the number of layers in images 6 through 9 in order to send more layers of image 10. The reason for this is that the algorithm delivers some of the layers in images 5 and 10 late, which then contribute less than a full unit towards the quality of their images, so the algorithm works harder to send additional layers of these two objects. Image 10 dominates the transmission policy, because it is very large in size (almost twice as large as the next largest image) and it is preceded by a very short interval (half as long as the next shortest interval).

Figure 13: Percentage of layers sent by the refined max-min algorithm with and without progressive rendering

8 Conclusion

As multimedia authorship tools develop in the coming years, synchronized multimedia presentations will likely become a popular medium for Web pages and e-mail messages. Because the Internet does not provide guarantees on QoS, it is desirable to introduce layered encoding into multimedia presentations, so that applications can maintain continuous play out and fully utilize available bandwidth to deliver a presentation of highest possible quality with minimum start up delay.

In this paper we have developed a comprehensive methodology for the streaming of layer-encoded multimedia presentations based on a two-phase decision problem. In phase one, the application transmits base layer data that comprise a presentation of minimum acceptable quality. When the application determines that enough bandwidth will most likely exist to deliver the remaining base layers on time, it starts play out of the presentation and transitions into phase two. In phase two, the application loops on the decision on which layer of which object to transmit next. To make this decision, we propose that the application continuously re-evaluate its expectation of future bandwidth, and based on this, compute an optimal transmission policy that maximizes an overall measure of presentation quality. We proposed two different presentation quality metrics: the *total quality* metric, which yields a decision problem that can be solved with dynamic programming, and second, the *refined max-min* metric that yields a computationally inexpensive optimal algorithm.

Much work remains to be done in the area. From a practical perspective, layered compression schemes for audio and video need to be further developed, and standards developers need to consider how existing protocols and document languages may need to be modified for layer encoded presentations. Also, work is needed in studying methods of estimating a lower bound on future bandwidth, whose accuracy improves over time following the initial connection between client and server. From a theoretical perspective, optimization criteria with progressive rendering need to be studied in greater detail. We are considering all of these issues in our ongoing research activities, as well as developing client/server implementations of the optimal policies.

9 Appendix

Proof of Theorem 1 Let $f(s,i)$ be defined as in (2), and let $g(s,i)$ be defined as in (4) and (5). We use an inductive argument to show that f and g are identical. For the case of $i = M$, note that since $\Omega_M(s) = \Big\{(j) \mid y_M(j) \leq s + b_M\Big\}$ and $\mathcal{X}_i(s) = \Big\{j \mid y_M(j) \leq s + b_M\Big\}$, j ranges through the same values in the maximizations of $q_M(j)$ over $\Omega_M(s)$ in (2) and $\mathcal{X}_M(s)$ in (4). Thus,

$$f(M,s) = \max_{\Omega_M(s)} \Big(q_M(j_M)\Big) = \max_{\mathcal{X}_M(s)} \Big(q_M(j)\Big) = g(M,s).$$

This establishes equality for the case $i = M$.

Now, we make the inductive assumption that $g(i+1,s) = f(i+1,s)$. We demonstrate that $g(i,s) = f(i,s)$ by first showing $g(i,s) \geq f(i,s)$, and then $g(i,s) \leq f(i,s)$. By definition of g as satisfying (5), we have

$$g(i,s) = \max_{j \in \mathcal{X}_i(s)} \Big(q_i(j) + g\Big(i+1, s + b_i - y_i(j)\Big)\Big).$$

Combining this with our inductive assumption, we have

$$g(i,s) = \max_{j \in \mathcal{X}_i(s)} \Big(q_i(j) + f\Big(i+1, s + b_i - y_i(j)\Big)\Big),$$

and then by definition of f in (2),

$$g(i,s) = \max_{\mathcal{X}_i(s)} \Big(q_i(j) + \max_{\Omega_{i+1}(s+b_i-y_i(j))} \Big(q_{i+1}(j_{i+1}) + \ldots + q_M(j_M)\Big)\Big) \tag{6}$$

Suppose that $(j_i(s), \ldots, j_M(s))$ maximizes $q_i(j_i) + \ldots + q_M(j_M)$ over $\Omega_i(s)$. Then by definition of f in (2), we have

$$f(i,s) = q_i\Big(j_i(s)\Big) + \ldots + q_M\Big(j_M(s)\Big). \tag{7}$$

Additionally, the first inequality of (1), which defines $\Omega_i(s)$, gives

$$y_i\Big(j_i(s)\Big) \le s + b_i.$$

Thus, $\mathcal{X}_i(s)$, as defined in (3), contains $j_i(s)$. This means the outer maximization in (6) covers $j_i(s)$. Thus we have:

$$g(i,s) \ge q_i\Big(j_i(s)\Big) + \max_{\Omega_{i+1}(s+b_i-y_i(j_i(s)))} \Big(q_{i+1}(j_{i+1}) + \ldots + q_M(j_M)\Big) \tag{8}$$

Substituting $j_i(s), \ldots, j_M(s)$ into the other inequalities in (1) (omitting the first inequality) produces the following system:

$$\begin{aligned}
&y_i\Big(j_i(s)\Big) + y_{i+1}\Big(j_{i+1}(s)\Big) \le s + b_i + b_{i+1}\\
&y_i\Big(j_i(s)\Big) + y_{i+1}\Big(j_{i+1}(s)\Big) + y_{i+2}\Big(j_{i+2}(s)\Big) \le s + b_i + b_{i+1} + b_{i+2}\\
&\vdots\\
&y_i\Big(j_i(s)\Big) + y_{i+1}\Big(j_{i+1}(s)\Big) + \ldots + y_M\Big(j_M(s)\Big) \le s + b_i + b_{i+1} + \ldots + b_M
\end{aligned}$$

Shifting the first term $y_i\Big(j_i(s)\Big)$ onto the right side of these inequalities gives us the system:

$$\begin{aligned}
&y_{i+1}\Big(j_{i+1}(s)\Big) \le s + b_i - y_i\Big(j_i(s)\Big) + b_{i+1}\\
&y_{i+1}\Big(j_{i+1}(s)\Big) + y_{i+2}\Big(j_{i+2}(s)\Big) \le s + b_i - y_i\Big(j_i(s)\Big) + b_{i+1} + b_{i+2}\\
&\vdots\\
&y_{i+1}\Big(j_{i+1}(s)\Big) + \ldots + y_M\Big(j_M(s)\Big) \le s + b_i - y_i\Big(j_i(s)\Big) + b_{i+1} + b_M
\end{aligned}$$

By definition of Ω, this system implies that $\Big(j_{i+1}(s) \ldots, j_M(s)\Big) \in \Omega_{i+1}\Big(s + b_i - y_i\Big(j_i(s)\Big)\Big)$. Therefore, the maximization in (8) covers the point $\Big(j_{i+1}(s), \ldots, j_M(s)\Big)$, and so we have from (8) and (7) that

$$g(i,s) \ge q_{i+1}\Big(j_{i+1}(s)\Big) + \ldots + q_M\Big(j_M(s)\Big) = f(i,s).$$

Now we complete the proof by showing $g(i,s) \le f(i,s)$. For this purpose, suppose that $j_i(s)$ is an element in $\mathcal{X}_i(s)$ that maximizes the outer maximization in (6). Then we can rewrite g as follows:

$$g(i,s) = q_i\Big(j_i(s)\Big) + \max_{\Omega_{i+1}(s+b_i-y_i(j_i(s)))} \Big(q_{i+1}(j_{i+1}) + \ldots + q_M(j_M)\Big)\Big). \tag{9}$$

Since $j_i(s) \in \mathcal{X}_i(s)$, we have

$$y_i\Big(j_i(s)\Big) \le s + b_i. \tag{10}$$

Now suppose that $\Big(j_{i+1}(s), \ldots, j_M(s)\Big)$ solves the maximization in (9), so that

$$g(i,s) = q_i\Big(j_i(s)\Big) + \ldots + q_M\Big(j_M(s)\Big), \tag{11}$$

and

$$\Big(j_{i+1}(s), \ldots, j_M(s)\Big) \in \Omega_{i+1}\Big(s + b_i - y_i\Big(j_i(s)\Big)\Big). \tag{12}$$

Statement (12) implies the following system of inequalities:

$$\begin{aligned}
&y_{i+1}(j_{i+1}(s)) \le s + b_i - y_i(j_i(s)) + b_{i+1} \\
&y_{i+1}(j_{i+1}(s)) + y_{i+2}(j_{i+2}(s)) \le s + b_i - y_i(j_i(s)) + b_{i+1} + b_{i+2} \\
&\vdots \\
&y_{i+1}(j_{i+1}(s)) + \ldots + y_M(j_M(s)) \le s + b_i - y_i(j_i(s)) + b_{i+1} + \ldots + b_M
\end{aligned} \tag{13}$$

If we shift the term $-y_i(j_i(s))$ appearing in each inequality to the left side, we see that (10) and (13) imply that $(j_i(s), \ldots, j_M(s))$ satisfies (1), and thus is a member of $\Omega_i(s)$. Therefore, $(j_i(s), \ldots, j_M(s))$ is covered by the maximization in (2) that defines f, and so we have

$$f(i,s) \ge q_i\Big(j_i(s)\Big) + \ldots + q_M\Big(j_M(s)\Big).$$

This result combined with (11) gives us $f(i,s) \ge g(i,s)$. ■

Claim The algorithm in Fig. 3 produces an optimal policy for the refined max-min criterion. After establishing the following lemma, we proceed to the main proof of this claim.

Lemma 1 *When the algorithm removes an object k from S, we have that*
$q_i(j_i - 1) < q_k(j_k)$ for $i \in S$.

Proof Suppose we are at the point in time when the algorithm removes k from S. Relative to this point, let i be an arbitrary element in S, and let j'_k and j'_i be the number of layers assigned to objects k and i, respectively. We demonstrate the lemma by showing:

$$q_i(j'_i - 1) < q_k(j'_k). \tag{14}$$

Case 1 $j'_i = 1$.

It is easy to see that (14) holds by observing that $q_i(j'_i - 1) = q_i(0) = 0$, and $q_k(j'_k) \ge q_k(1) > 0$.

Case 2 $j'_i > 1$.

Consider the point prior to the removal of k from S when the algorithm last incremented the layer counter for object i, that is, when j_i was equal to $j'_i - 1$. Relative to this point, let j''_k be the number of layers assigned to object k. Because the layer counter for object i was being incremented, the quality of object i was less than the quality of all objects in S, and in particular, the quality of object k. Thus,

$$q_i(j'_i - 1) < q_k(j''_k). \tag{15}$$

Since this time precedes the point when k is removed from S, we must have that $j''_k \leq j'_k$, which implies, since q is increasing, that

$$q_k(j''_k) \leq q_k(j'_k). \tag{16}$$

Taken together, (15) and (16) demonstrate (14). ■

Proof of Claim Let $\widehat{P} = (\widehat{j}_1, \ldots, \widehat{j}_M)$ be an optimal policy, that is, a feasible policy whose quality is greater than or equal to all other feasible policies. Let $\widehat{a}_i$ represent the object with the i^{th} best quality under $\widehat{P}$, which means that the i^{th} component of $Q(\widehat{P})$ is $q_{\widehat{a}_i}(\widehat{j}_{\widehat{a}_i})$. At an arbitrary point during the execution of the algorithm, let $P = (j_1, \ldots, j_M)$ represent the current policy, and a_i the object with the i^{th} best quality under P. Thus, the i^{th} component of $Q(P)$ is $q_{a_i}(j_{a_i})$. Each time an object is removed from S, its assigned layers (and thus its quality) becomes fixed. At the point when the m^{th} object is removed from S, the following inequalities will hold for the duration of the algorithm:

$$q_{a_1}(j_{a_1}) < q_{a_2}(j_{a_2}) < \ldots < q_{a_i}(j_{a_m})$$

Additionally, since the objects remaining in S have qualities higher than $q_{a_m}(j_{a_m})$, and since the number of layers assigned to them are never decreased, we must have that the following holds true after the m^{th} removal from S:

$$q_{a_m}(j_{a_m}) < q_{a_k}(j_{a_k}) \ \text{ for } \ k = m+1, \ldots, M$$

We prove the theorem inductively. In step 1, we demonstrate that when the algorithm removes its first object from S, we have

$$q_{a_1}(j_{a_1}) = q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1}).$$

In step 2, we make the inductive assumption that after m removals from S,

$$q_{a_i}(j_{a_i}) = q_{\widehat{a}_i}(\widehat{j}_{\widehat{a}_i}) \ \text{ for } \ i = 1, \ldots, m, \tag{17}$$

and then show that when the algorithm removes its next object from S, the following must hold:

$$q_{a_{m+1}}(j_{a_{m+1}}) = q_{\widehat{a}_{m+1}}(\widehat{j}_{\widehat{a}_{m+1}}).$$

Thus, when the algorithm removes the M^{th} object from S and terminates, we have that $Q(P) = Q(\widehat{P})$.

Before beginning step 1, note that since the quality values are assumed to be unique across all layers and objects within the presentation, (17) is equivalent to the following:

$$a_i = \widehat{a}_i \text{ and } j_{a_i} = \widehat{j}_{\widehat{a}_i} \text{ for } i = 1, \ldots, m \tag{18}$$

Therefore, we have as a corollary to the theorem that the optimal refined max-min policy is unique.

Step 1 *When the algorithm removes its first object from S, we have that $q_{a_1}(j_{a_1}) = q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$.*

Suppose the algorithm is removing its first object from S. As we noted before, this object will be permanently assigned to a_1, and it will have lowest quality in all remaining policies computed by the algorithm. We demonstrate equality by showing that both $q_{a_1}(j_{a_1}) > q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$ and $q_{a_1}(j_{a_1}) < q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$ lead to contradictions.

Assume that $q_{a_1}(j_{a_1}) > q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$. Because a_1 has been removed from S, all other objects have qualities greater than $q_{a_1}(j_{a_1})$ under P, which means that $q_{a_i}(j_{a_i}) > q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$ for $i = 1, \ldots, M$. Thus, $Q(P) > Q(\widehat{P})$, which contradicts the fact that $\widehat{P}$ is optimal.

Now assume $q_{a_1}(j_{a_1}) < q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$. Since $q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$ is less than all other values of $q_{\widehat{a}_i}(\widehat{j}_{\widehat{a}_i})$, we have

$$q_{a_1}(j_{a_1}) < q_{\widehat{a}_i}(\widehat{j}_{\widehat{a}_i}) \text{ for } i = 1, \ldots, M.$$

Since $\{\widehat{a}_1, \ldots, \widehat{a}_M\} = \{1, \ldots, M\}$, the above is equivalent to

$$q_{a_1}(j_{a_1}) < q_i(\widehat{j}_i) \text{ for } i = 1, \ldots, M. \tag{19}$$

Since $a_1 \in \{1, \ldots, M\}$, (19) gives $q_{a_1}(j_{a_1}) < q_{a_1}(\widehat{j}_{a_1})$. Because q is increasing, we must have that $j_{a_1} < \widehat{j}_{a_1}$, and then, since j_i takes only integer values,

$$j_{a_1} + 1 \leq \widehat{j}_{a_1}. \tag{20}$$

Combining our assumption that $q_{a_1}(j_{a_1}) < q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$ with the lemma, we have

$$q_i(j_i - 1) < q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1}) \text{ for } i \in S.$$

Since $q_{\widehat{a}_1}(\widehat{j}_{\widehat{a}_1})$ is less than all other values of $q_{\widehat{a}_i}(\widehat{j}_{\widehat{a}_i})$, we can state that $q_i(j_i - 1) < q_i(\widehat{j}_i)$, which leads to $j_i - 1 < \widehat{j}_i$, and then

$$j_i \leq \widehat{j}_i \text{ for } i \in S. \tag{21}$$

Taken together, (20) and (21) imply that $(j_1, \ldots, j_{a_1} + 1, \ldots, j_M)$ is component-by-component less than $\widehat{P}$, and therefore must be feasible. But this contradicts the fact that a_1 is being removed from S.

Step 2 *If the first m removals from S result in (17), then the next removal from S results in $q_{a_{m+1}}(j_{a_{m+1}}) = q_{\widehat{a}_{m+1}}(\widehat{j}_{\widehat{a}_{m+1}})$.*

Similar to step 1, we show that $q_{a_{m+1}}(j_{a_{m+1}}) > q_{\widehat{j}_{\widehat{a}_{m+1}}}(\widehat{a}_{m+1})$ and $q_{a_{m+1}}(j_{a_{m+1}}) < q_{\widehat{a}_{m+1}}(\widehat{j}_{\widehat{a}_{m+1}})$ lead to contradictions.

Assume that $q_{a_{m+1}}(j_{a_{m+1}}) > q_{\widehat{j}_{\widehat{a}_{m+1}}}(\widehat{a}_{m+1})$. Since the objects remaining in S have qualities greater than $q_{a_{m+1}}(j_{a_{m+1}})$, we have that $q_{a_i}(j_{a_i}) > q_{\widehat{a}_{m+1}}(\widehat{j}_{\widehat{a}_{m+1}})$ for $i = m+1, \ldots, M$. Thus, $Q(P) > Q(\widehat{P})$, which contradicts the assumption that $\widehat{P}$ is optimal.

Now, assume that $q_{a_{m+1}}(j_{a_{m+1}}) < q_{\widehat{a}_{m+1}}(\widehat{j}_{\widehat{a}_{m+1}})$. Because $q_{\widehat{a}_k}(\widehat{j}_{\widehat{a}_k})$ is increasing in k, we have that

$$q_{a_{m+1}}(j_{a_{m+1}}) < q_i(\widehat{j}_i) \text{ for } i \in \{\widehat{a}_{m+1}, \ldots, \widehat{a}_M\} \tag{22}$$

Because our inductive assumption is equivalent to (18), we have that $\{a_1, \ldots, a_m\} = \{\widehat{a}_1, \ldots, \widehat{a}_m\}$, and then,

$$\{a_{m+1}, \ldots, a_M\} = \{\widehat{a}_{m+1}, \ldots, \widehat{a}_M\}. \tag{23}$$

Thus, 22 holds for $i \in \{a_{m+1}, \ldots, a_M\}$, and in particular, for $i = a_{m+1}$. Therefore, we have that $q_{a_{m+1}}(j_{a_{m+1}}) < q_{a_{m+1}}(\widehat{j}_{a_{m+1}})$. Because q is strictly increasing, we have $j_{a_{m+1}} < \widehat{j}_{a_{m+1}}$, and then

$$j_{a_{m+1}} + 1 \leq \widehat{j}_{a_{m+1}}. \tag{24}$$

The lemma guarantees that $q_i(j_i - 1) < q_{a_{m+1}}(j_{a_{m+1}})$ for $i \in S = \{a_{m+2}, \ldots, a_M\}$. But this is also true for $i = a_{m+1}$, because q is increasing. Combining this with our assumption that $q_{a_{m+1}}(j_{a_{m+1}}) < q_{\widehat{a}_{m+1}}(\widehat{j}_{\widehat{a}_{m+1}})$, we get

$$q_i(j_i - 1) < q_{\widehat{a}_{m+1}}(\widehat{j}_{\widehat{a}_{m+1}}) \text{ for } i \in \{a_{m+1}, \ldots, a_M\}. \tag{25}$$

Because $q_{\widehat{a}_k}(\widehat{j}_{\widehat{a}_k})$ is increasing in k, and because of (23) we have that

$$q_i(j_i - 1) < q_{\widehat{a}_k}(\widehat{j}_k) \text{ for } i \in \{a_{m+1}, \ldots, a_M\} \text{ and } k \in \{\widehat{a}_{m+1}, \ldots, \widehat{a}_M\}$$

But since $\{a_{m+1}, \ldots, a_M\}$ and $\{\widehat{a}_{m+1}, \ldots, \widehat{a}_M\}$ are identical, we have that $q_i(j_i - 1) < q_i(\widehat{j}_i)$ for $i \in \{a_{m+1}, \ldots, a_M\}$, and thus $j_i - 1 < \widehat{j}_i$, for $i \in \{a_{m+1}, \ldots, a_M\}$. This implies

$$j_i \leq \widehat{j}_i \text{ for } i \in \{a_{m+2}, \ldots, a_M\}. \tag{26}$$

Equations (18), (24) and (26) taken together imply that $(j_1, \ldots, j_{a_{m+1}} + 1, \ldots, j_M)$ is component-by-component less than or equal to $\widehat{P}$, and therefore must be feasible. But this contradicts the fact that a_{m+1} is being removed from S. ■

References

[1] E. Denardo. Dynamic Programming. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[2] D. Turner and K. Ross. Optimal Streaming of Layer-Encoded Multimedia Presentations. Proceedings of the IEEE Interenational Conference on Multimedia and Expo, New York, NY, Jul/Aug, 2000.

[3] D. Turner. Asynchronous Multimedia Messaging. Ph.D. Thesis, Eurecom Institut, Jun 2001.

[4] D. Turner. Optimal Streaming of Stored Scalable Media. Proceedings of the Multimedia Technologies and Applications Conference, University of California, Irvine, CA, Nov 2001.

[5] P. Hoschka, ed. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, W3C Recommendation, Jun 1998.

[6] W. Zhao, M. Willebeek-LeMair and P. Tiwari. Malleable Multimedia Presentations: Adaptive Streaming Tradeoffs for Best-Quality Fast-Response Systems. Proc. of the 10th Tyrrhenian International Workshop on Digital Communications, Sep 1998.

[7] W. Zhao, M. Willebeek-LeMair and P. Tiwari. Efficient Adaptive Media Scaling and Streaming of Layered Multimedia in Heterogeneous Environments. Proc. of the IEEE International Conference on Multimedia Computing and Systems, Jun 1999.

[8] K.S. Candan, B. Prabhakaran and V.S. Subrahmanian. Retrieval Schedules Based on Resource Availibility and Flexible Presentation Specifications. Multimedia Systems, July, 1998.

[9] D. Saparilla and K. Ross. Streaming Stored Continuous Media over Fair-Share Bandwidth. Eurecom technical report, Feb 2000.

[10] W. Feng. Video-on-Demand Services: Efficient Transportation and Decompression of Variable Bit Rate Video. Ph.D. Dissertation, April 1996.

[11] J. McManus and K. Ross. Video on Demand over ATM: Constant-Rate Transmission and Transport. IEEE JSAC, Vol. 14, 1996.

[12] J. McManus and K. Ross. A Dynamic Programming Methodology for Managing Layered Encoded VBR Sources in Packet-Switched Networks. Telecommunications Systems, Vol. 9, 1998.

[13] J. Salehi, Z. Zhang, J. Kurose and D. Towsley. Supporting Stored Video: Reduce Rate Variability and End-to-End Resource Requirements through Optimal Smoothing. Proceedings of ACM SIGMETRICS, 1996.

[14] J. Salehi, Z. Zhang, J. Kurose and D. Towsley. Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing. IEEE/ACM Transactions on Networking, Aug 1998.

[15] M. Reisslein and K. Ross. Join-the-Shortest-Queue Prefetching. ICNP, Atlanta, 1997.

[16] M. Reisslein and K. Ross. High Performance Prefetching Protocols for VBR Prerecorded Video. IEEE Network Magazine, Nov/Dec 1998.

[17] Independent JPEG Group. JPEG software library, availble at http://www.jcu.edu.au/docs/jpeg.

[18] T. Ibaraki and N. Katoh. Resource Allocation Problems: Algorithmic Approaches. MIT Press, 1988.

[19] H. Luss and D.R. Smith. Resource allocation amoung competing activities: a lexigographic minimax approach, Operation Research Letters, 5(1986), 227-231.

[20] Z. Cao and E. Zegura. Utility max-min: an application-oriented bandwidth allocation scheme. Proceedings of IEEE INFOCOM 99, New York, NY, Mar 1999.

[21] Xiphophorus. Ogg Vorbis audio codec under the LGPL. (www.vorbis.org)