Optimal Streaming of Synchronized Multimedia Presentations with Layered Objects

David A. Turner and Keith W. Ross
Institut Eurécom
B.P. 193, 06904 Sophia-Antipolis, France
{turner, ross}@eurecom.fr

Abstract -- A synchronized 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. In this paper we consider the problem of streaming a multimedia presentation from a server to a client over a bandwidth-limited communication network. We suppose that each of the static objects is layered-encoded. For a given maximum delay, we consider the problem of finding the optimal number of layers in each object in order to maximize a measure of the overall quality of the presentation. We devise efficient algorithms for determining an optimal policy for several natural criteria. We also consider the problem of gradual rendering of objects after their start times. We then apply the algorithms to a randomly generated presentation containing layer-encoded JPEG images.

1 Introduction

A synchronized 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), animation, formatted text (such as HTML), etc. The Synchronized Multimedia Integration Language (SMIL) [11] is an example of a presentation description language that describes the temporal and spatial layout of presentation objects.

In this paper we consider the problem of streaming a multimedia presentation from a server to a client over a bandwidth-limited communication network. We assume that bandwidth available to the multimedia presentation is constant and known; for example, the available bandwidth might be the average transmission rate of the client's modem connection, which is often the bottleneck bandwidth between the server and the client.

It is important to make the distinction between the continuous-media (CM) objects and the static objects that make up a multimedia presentation. CM objects include audio, video and other objects whose bits are continuously processed over a playback interval. Static objects include images (such as JPEGs and GIFs), formatted text and other objects whose bits are rendered at discrete times. CM streams are typically controlled by a streaming protocol such as RTSP according to the temporal layout of the presentation [13]. With today's popular streaming products (e.g., RealNetworks' streaming products [15]), the server transmits each CM stream at the instantaneous consumption rate of the stream, and the client renders each CM stream as it receives the stream. These products introduce a small playback delay in order to build a small buffer to remove network jitter. Our model reflects the trends in CM streaming in that it assumes the transmission rate of a CM stream is equal to its consumption rate.

Because static objects must be present in their entirety before their first appearance in the presentation, the client must retrieve each static object prior to its scheduled appearance, which we call its deadline. At any instant of time, the bandwidth that is available for the transmission of the static objects is the network bandwidth minus the bandwidth that is being consumed by the ongoing CM streams. The static objects can fully utilize this available bandwidth. The server application can explicitly match the transmission rate of the static objects to the available bandwidth; or the server can implicitly match the transmission rate to the available bandwidth by sending the static objects over a self-regulating transport protocol such as TCP.

In our model we suppose that each of the static objects is layered-encoded. For example, each of the static objects could be a progressively encoded JPEG image [14]. For a layered-encoded object, rendering a layer requires all of the lower layers to be present at the client. When a presentation has layered objects, the server has the option of not transmitting some of the layers for some of the objects in order to reduce start-up latency. However, the overall quality of the presentation is reduced when layers are dropped. Thus there is a critical tradeoff between the quality of the presentation and the start-up latency. For a given start up delay, we consider the problem of finding the optimal number of

layers in each object in order to maximize a measure of the overall quality of the presentation.

We allow each layer level of each static 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, the mean squared error of the image rendered, or the number of layers rendered times the fraction of time the layers are displayed in the presentation.

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 maximizes the total quality for a fixed start-up delay. But for many natural quality functions for the individual objects, the total quality criterion has a tendency to overly emphasize objects that are near the end of the presentation. To avoid this result, we introduce a new criterion that we refer to as the refined max-min criterion. The refined max-min criterion strives to equalize the quality values of all the static 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.

This paper is organized as follows. In Section 2 we introduce the model for the case when all static objects have a single layer. In Section 3 we introduce layered static objects and discuss the tradeoff between presentation quality and start-up delay. In Section 4 we present the refined max-min criterion and an efficient algorithm for its solution. In Section 5 we present the total quality criterion and show how it can be formulated as a dynamic programming equation. In Section 6 we use data from progressive JPEG images to compare the optimization criteria for natural definitions of object quality. In section 7 we consider gradual rendering. In Section 8 we review related research. We conclude in Section 9.

2 Minimizing the Start Up Latency: The Single Layer Case

In this section we briefly consider the case of one layer for all static objects, i.e., no layer encoding. This allows us to define notation and set the stage for the more interesting case of layered objects.

Today (June 1999) household PCs are sold with 10 gigabytes of disk storage, and disk storage has been doubling every year. Therefore, for the vast majority of Internet users worldwide, client storage capacity is not a constraint, and can be safely assumed to be limitless. One the other hand, access bandwidth remains a constraining resource. Although xDSL and cable modems have been recently introduced, the majority of Internet users remain behind dial-up and ISDN modems. The transmission rates for the dial-up modem and ISDN access links range between 28 Kbps and 128 Kbps, and are typically the bottleneck transmission rates between server and client. It is therefore reasonable to assume (as a first-order approximation) that the available bandwidth from the server to the client, averaged over a few RTTs, is nearly constant. Let B denote this average available bandwidth.

Because the CM data is being rendered approximately within the instant in which it arrives at the client, a predetermined amount of bandwidth will be consumed by the CM flows at each point in time. Thus at each point in time, we can partition the total bandwidth B into that which is being used by the CM flows and that which is available for the transport of the static data. Our first step is to develop an expression for the cumulative bandwidth claimed by the CM flows.

Suppose there are M CM flows in the presentation and that $[a_m, b_m]$ is the rendering interval for the m^{th} CM flow. All expressions of time in this paper will be relative to the presentation timeline, where time $t = 0$ represents the moment in which the presentation begins rendering at the client. In order to remove the effects of network jitter, it is necessary to buffer a small amount of the stream prior to commencing playback. Let α represent the number of seconds that are used to build up the jitter buffer. Thus transmission of the m^{th} CM flow will begin at time $a'_m = a_m - \alpha$. These bits will be transmitted at a constant rate r_m until the last bit of the stream arrives. The transmission of the stream ends at time $b'_m = b_m - \alpha$. The cumulative number of bits of bandwidth consumed by the m^{th} flow has the following form:

$$y_m^{CM}(t) = \begin{cases} 0 & t < a'_m \\ (t - a'_m) \cdot r_m & a'_m \le t \le b'_m \\ (b'_m - a'_m) \cdot r_m & b'_m < t \end{cases}$$

The cumulative bandwidth claimed by all the CM flows is the sum of the cumulative bandwidth claimed by the individual flows:

$$B^{CM}(t) = \sum_{m=1}^{M} y_m^{CM}(t)$$

If we begin transmitting presentation data d seconds prior to rendering (i.e., at time $t = -d$), then the maximum number of bits that can be received at the client by time t is $B \cdot (t + d)$. Thus the cumulative bandwidth available to the static objects up to time t is:

$$B_d^{static}(t) = B \cdot (t + d) - B^{CM}(t) \qquad (1)$$

In addition to the m CM flows, let N denote the number of static objects, where the i^{th} object has y_i bits and initial rendering time t_i. Without loss of generality we can assume that the N objects are ordered in their presentation orders. We refer to the rendering times as arrival deadlines, because the bits of each static object must arrive before these times in order that rendering occurs on schedule. For each deadline t_i, we must have that the bits of objects 1 through i are not greater than the cumulative bits available for their transmission, as follows:

$$\sum_{i=1}^{k} y_i \le B_d^{static}(t_k) \quad \text{for} \quad k = 1,...,N \qquad (2)$$

Combining expressions (1) and (2), and rearranging, we obtain the minimum start up delay for the presentation:

$$d_{\min} = \max_{1 \le k \le N} \left\{ \frac{\sum_{i=1}^{k} y_i + B^{CM}(t_k)}{B} - t_k \right\}$$

3 Multimedia Presentations with Layered Static Objects

For the remainder of this paper we suppose that all objects are layered-encoded. Such layered objects can be rendered in increasing degrees of quality as more layers of the object are used. For a given maximum start-up delay, we consider the problem of determining which layers to transmit in each of the objects so that presentation quality is optimal. Let d denote the maximum start up delay, which we consider given and fixed throughout the remainder of this paper.

As before, we let t_i represent the deadline for object i. Also, we denote the time transmission commences by t_0, i.e., $t_0 = -d$. Let b_i be the number of bits that can be transmitted between the deadlines t_{i-1} and t_i, i.e., $b_i = B^{static}(t_i) - B^{static}(t_{i-1})$. Henceforth we refer to the interval $[t_{i-1}, t_i]$ as the i^{th} interval, and b_i as the number of bits (from static objects) that can be transferred in this interval to the client.

We use L_i to represent the number of layers of object i, and x_{ij} to represent the number of bits in its j^{th} layer. We define $y_i(j)$ as the cumulative bits required to send layers 1 through j of object i, i.e.

$$y_i(j) = x_{i1} + ... + x_{ij}.$$

A delivery policy specifies the number of layers to send for each object, which we represent as an N-dimensional vector P whose i^{th} component j_i represents the number of layers of object i to send to the client. That is, $P = (j_1, \ldots, j_N)$, where $j_i =$ number of layers to send of object i.

Policy $(L_1, \ldots, L_N)$ will obviously result in the highest quality presentation, but it may not be possible to send this amount of data under the given bandwidth and delay constraints. We call a policy *feasible* if all of the bits sent arrive at the client prior to their deadlines. Obviously we should send bits with earlier deadlines before bits with later deadlines; thus we can express feasibility with the following system of N inequalities:

$$\begin{aligned} & y_1(j_1) \le b_1 \\ & y_1(j_1) + y_2(j_2) \le b_1 + b_2 \\ & \quad \vdots \\ & y_1(j_1) + \ldots + y_N(j_N) \le b_1 + \ldots + b_N \end{aligned}$$

In words, at each deadline, the cumulative number of bits needed must be less than or equal to the cumulative bits that could have been transmitted.

Now that we have defined the set of feasible policies (which result in uninterrupted presentations), we need to determine which of these policies results in presentations with the greatest quality. To this end, we

say that object i achieves quality value $q_i(j)$ if layers 1 through j of object i are displayed. These quality values can be defined by subjective evaluation, by quantitative measures (such as mean squared error) or by simple natural choices. For example, a natural choice is $q_i(j) = j/L_j$,in which the quality of an image is simply equal to the percentage of the object's layers that are rendered. Another natural measure is $q_i(j) = y_i(j)/y_i(L_i)$, which associates the quality value with the fraction of bits rendered. We also might want to assign relatively more weight to objects at the beginning of the presentation (in order to lock in the user's attention) or more weight to the bottom layers of the images (in order to encourage homogeneity in quality across images). We also may want to determine the quality of an object by taking into account the length of time the object is displayed. Clearly, a general function $q_i(j)$ provides great flexibility in defining quality, and should allow us to match the model's definition of quality to realistic values. Naturally we assume that $q_i(j)$ increases as the number of layers j increases.

In order to determine an optimal policy P, we need to define the overall quality of a presentation. One natural definition of presentation quality is the sum of the quality values of the individual static objects, which we call its *total quality*. We consider optimizing the total quality in Section 5. In the next section we introduce a criterion which strives to balance the quality values of the objects while using the available bandwidth to its fullest.

4 The Refined Max-min Criterion

A natural definition of the overall quality of a presentation is the worst quality of its objects, which we call the *max-min criterion*:

$$Q(P) = \min_{\{(j_1,\ldots,j_N)\,|\,j_i \le L_i\}} \{q_1(j_1),\ldots,q_N(j_N)\}.$$

Although the max-min criterion is a natural criterion, 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, if the target start up delay is small and the first image is to be rendered at the beginning of the presentation, it will be possible to send only a few layers of the first object, which then becomes the minimum quality set for all images in the presentation. In general, less bandwidth is 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.

We refine our notion of the overall quality of a presentation by representing it with a vector rather than a scalar value, and define an ordering of these vectors. Let $A = (j_1,\ldots,j_N)$ and $B = (k_1,\ldots,k_N)$ be two policies. These will have corresponding quality vectors,

$$Q(A) = (q_1(j_1),\ldots,q_N(j_N)), \text{ and}$$
$$Q(B) = (q_1(k_1),\ldots,q_N(k_N)).$$

Now sort the elements in each of these vectors from lowest to highest quality value to obtain their sorted quality vectors $(a_1,\ldots,a_N)$ and $(b_1,\ldots,b_N)$, respectively. We say that policy A has better quality than policy B if the sorted quality vectors are equal until some position k where the quality value of A is greater than that of B. That is, $A > B$ if there exists a k such that $a_i = b_i$ for $i < k$, but that $a_k > b_k$. A feasible policy A is said to be optimal if there does not exist another feasible policy B such that $B > A$. We refer to this criterion as the *refined max-min criterion*.

We clarify the refined max-min criterion with an example. Suppose that policy $A = (5,6,7,6,7,8)$, that is, under policy A, 5 layers of the first object are sent, 6 layers of the second, etc. Suppose that policy $B = (6,5,6,6,8,8)$. If we use the identity quality measure $q_i(j) = j$, then the quality vectors will be identical to the policy vectors, that is, $Q(A) = A$ and $Q(B) = B$. The sorted quality vectors will be $S_A = (5,6,6,7,7,8)$ and $S_B = (5,6,6,6,8,8)$. S_A and S_B match in the first three elements, but the fourth element of S_A, which is 7, is greater than the fourth element of S_B, which is 6. Therefore, policy A has better quality than policy B.

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 in general 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 or the number of bits typically fulfill this assumption. The algorithm (shown in Fig. 1) starts with the null policy and a set S consisting of all objects. It enters a loop in which it tries to add a layer to the object in S with minimum quality in S. If this results in a non-feasible policy, then it removes the object from S without adding a layer to it.

The set S is used to record which objects are not yet fixed in their number of layers in the final policy. If an object has minimum quality in the set S, but it is not feasible to improve its quality by sending an additional layer, then we no longer consider it by removing it from S. We terminate the loop when P includes all layers or when S becomes empty. The algorithm converges to an optimal policy.

In the case of continuous quality measures that are functions of the number of bits in a layer or the length of time the object is rendered, our assumption of distinct $q_i(j)$ is reasonable, and the above algorithm is adequate. However, for the quality measure $q_i(j) = j$, and other measures that map into a relatively small range, this algorithm may not converge to an optimal policy. If the quality values are not distinct, at some point within the loop there may be a non-singleton subset R of S whose elements have the minimum quality under policy P. If adding the next layer to P for all the elements of R results in a feasible policy, then there is no problem. However, it may be the case that a feasible policy may only be possible by adding

$P = (j_1, \ldots, j_N) = (0, \ldots, 0)$
$S = \{1, 2, \ldots, N\}$
do while $P \neq (L_1, \ldots, L_N)$ and 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 $(j_1, \ldots, j_k + 1, \ldots, j_N)$ is feasible then
    $j_k = j_k + 1$
  else
    remove k from S

Figure 1: Refined max-min algorithm

the next layer for a subset of R.

In Fig. 2, we present a modified version of the algorithm that converges to an optimal policy by properly resolving quality ties. However, for the algorithm to work, we must assume that when one object is larger than another object, the sizes of its layers are individually larger than the layers of the smaller object. We refer to this as the *consistency condition*. In notation,

$$x_{ij} > x_{kj} \qquad x_{in} > x_{kn} \quad \text{for } n > j$$

For each pass in the outer loop of the algorithm, we construct an ordered set (a vector) R of all objects from S that have the minimum quality value in relation to other objects of S. (Note that the elements of R should be in increasing order.) Then we enter an inner loop that passes forward through R. For each object in R, we add a layer in the policy vector P corresponding to that object. If this results in a non-feasible policy, then we must remove one layer either from this object or from an adequately sized earlier object of R. (By adequate, we mean that it would free up enough bits to transmit the last added layer.) At this point, the consistency condition has us drop the layer that consumes the greatest number of bits, because this will free up the greatest number of bits for the transmission of additional layers of the remaining objects in R, and for the transmission of additional layers in future passes through the outer loop. We terminate the loop when either the policy transmits all the data or S becomes empty. The algorithm produces an optimal policy when the consistency condition is

$S = \{1, \ldots, N\}$
$P = (j_1, \ldots, j_N) = (0, \ldots, 0)$
do while $P \neq (L_1, \ldots, L_N)$ and S is nonempty
  let $R = (i_1, \ldots, i_n)$ be the vector comprised of
    elements in S with minimum quality
  for k = 1 to n
    $j_{i_k} = j_{i_k} + 1$
    if P is not feasible then
      find $m \leq k$ s.t. object m is contributing
        the greatest number of bits
      remove m from R and S
      $j_{i_m} = j_{i_m} - 1$

Figure 2: Refined Max-min algorithm with tie-breaking

satisfied.

5 The Total Quality Criterion

For the total quality criterion, the quality of a presentation for policy $P=(j_1,\ldots,j_N)$, denoted by $Q(P)$, is sum of the individual quality values, that is,

$$Q(P)=q_1(j_1)+\ldots+q_N(j_N).$$

In this section we develop an efficient means to determine the policy P that maximizes the total quality $Q(P)$ subject to the constraint of uninterrupted playback. In particular, we show how to formulate this optimization problem as a dynamic programming (DP) problem.

Suppose that we want to send objects n through N, and that we have s surplus bits of bandwidth available to do this, in addition to the $b_n,\ldots,b_N$ bits that are available in the intervals terminating at deadlines $t_n,\ldots,t_N$. Define $f(n,s)$ to be the maximum quality attainable for these objects. That is,

$$f(n,s)=\max\{q_n(j_n)+\ldots+q_N(j_N)\},$$

where $j_n,\ldots,j_N$ satisfy the following constraints:

$$\begin{aligned} &y_n(j_n)\le s+b_n\\ &y_n(j_n)+y_{n+1}(j_{n+1})\le s+b_n+b_{n+1}\\ &\quad\vdots\\ &y_n(j_n)+\ldots+y_N(j_N)\le s+b_n+\ldots+b_N \end{aligned}$$

By definition, $f(1,0)$ is the maximum quality for the presentation.

Now consider the following functional equation of dynamic programming. For $n=1,\ldots,N-1$,

$$g(n,s)=\max_{\{j|y_n(j)\le s+b_n\}}\{q_n(j)+g(n+1,s+b_n-y_n(j))\},$$

and for $n=N$,

$$g(N,s)=\max_{\{j|y_N(j)\le s+b_N\}}\{q_N(j)\}.$$

It can be shown that $g(n,s)=f(n,s)$ for all n and s. (The proof is suppressed due to page limitation constraints.) Standard DP techniques can be employed to efficiently determine the optimal policy from these equations.

6 Experimental Results

We assembled a slide show presentation with images randomly taken from the WebMuseum, Paris and the University of Southern California Image Database, which includes 2 black and white and 8 color images. We encoded the images using the Independent JPEG Group's library [14], which by default encodes color images into 10 layers and black and white images into 6 layers. We chose rendering times that reflect a quick-paced slide-show presentation. Table 1 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. With this data we compare the transmission policies that result from using the total quality criterion and the refined max-min criterion for two different quality measures.

The first object quality measure is the ratio of layers rendered to the total number of layers in the object, i.e., $q_i(j)=j/L_i$. 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. Since the refined max-min

Table 1: Image Data for Slide-Show Presentation

Image	Deadline (secs)	Bytes by Layer Number										total
		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

algorithm relies on the consistency condition to break ties correctly, the policy that it produces may be sub-optimal when the condition is not met. In fact, we found the consistency condition to be weakly satisfied by the layered JPEG data that we investigated. (You can see this by examining Table 1.)

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:

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

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 are 36 Kbps of bandwidth available for the presentation, and that the audio stream consumes 12 Kbps of this total. We fix the start up delay at 5 seconds.

Fig. 3 shows the percentage of layers sent for each object for each of the two criteria, using 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 makes the weakest images slightly stronger and the strongest images slightly weaker. In other words, the refined max-min algorithm produces a presentation that is more uniform.

We can also see from Fig. 3 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. 4 shows the percentage of bits sent for each object when using the bit-oriented quality 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.

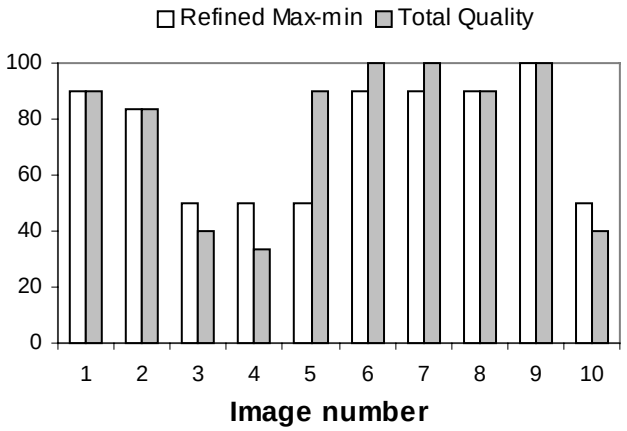

Figure 3: Percentage of layers sent by criterion under the layer-oriented quality measure

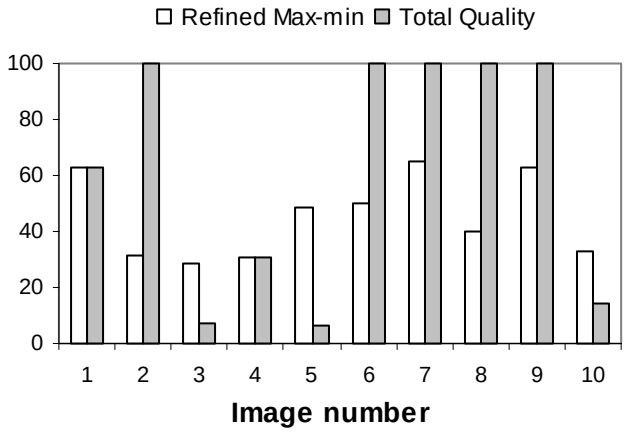

Figure 4: Percentage of bits sent by criterion under the bit-oriented quality measure

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: minimum percentage of layers rendered (Fig. 5), bandwidth utilization (Fig. 6), and average percentage of layers rendered (Fig 7).

Fig. 5 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 was 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-

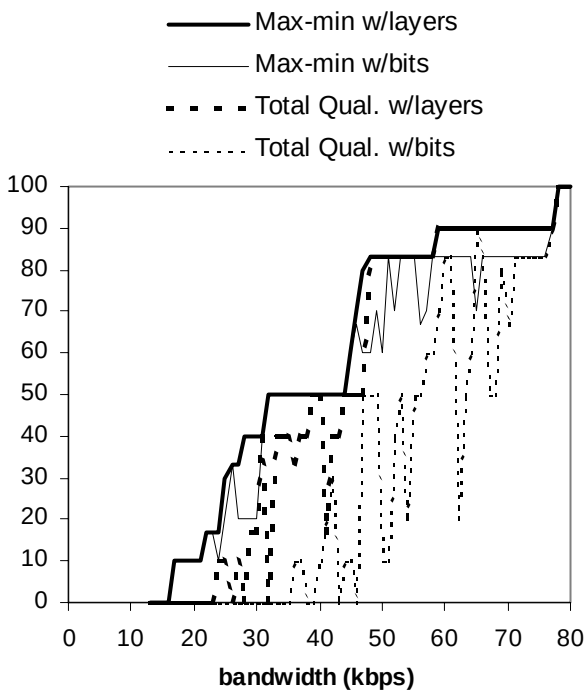

Figure 5: Minimum number of layers rendered by criterion/quality measure

oriented measure, but in general the refined max-min criterion is 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. 6 shows the bandwidth utilization (bandwidth 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 criteria works slightly better with the bit-oriented measure.

Fig. 7 shows the average of the percentage of layers rendered for each object. Here the total quality criterion based on the layer-oriented quality measure is superior to the other methods, especially for the lower 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 min-max criterion based on the layer-oriented quality measure converges the most quickly. The

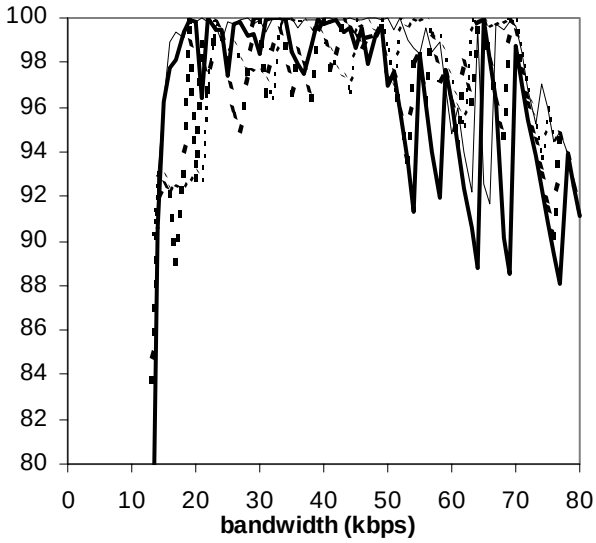

Figure 6: Bandwidth utilization by criterion/quality measure

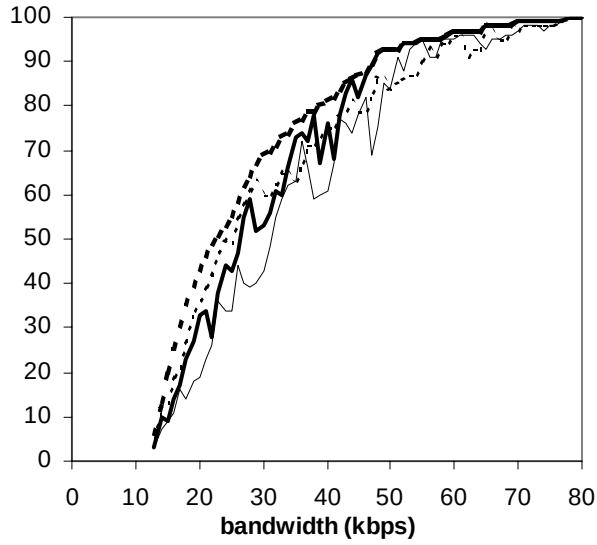

Figure 7: Minimum number of layers rendered by criterion/quality measure

refined max-min criterion with the bit-oriented quality measure is the clear loser in this comparison.

In summary, the four objectives can produce rather different optimal policies. We believe that the refined max-min criterion is superior to the total quality and max-min criteria. Nevertheless, in order to make a more definite conclusion, subjective testing with human subjects is needed. Also, it is desirable to experiment with other quality values $q_i(j)$ that take into account the mean-squared error.

7 GRADUAL RENDERING

Now we consider the benefits and methodology of gradual rendering of static presentation objects. In gradual 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 gradually rendered objects has value, consider a slide show presentation with an audio stream encoded at 12 Kbps, a start-up delay of three seconds, and a total bandwidth of 20 Kbps. This leaves 8 Kbps for the transmission of static content. Now suppose the first image is displayed for one minute, and that it contains 25 K bits in the first layer. Because the 3-second start up phase only permits the transfer of 24 K bits, the first layer can not fully arrive before the beginning of the object's rendering interval. Without permitting gradual rendering, no layers of the first image will be displayed, and the user's screen will be blank during the first one minute of the presentation.

In general, presentations with short start up delays will not be able to adequately render the initial object. But a small start up delay is important in many situations, thus a transmission policy that admits gradual rendering is needed.

Besides allowing shorter start up delays, another benefit of gradual rendering is to increase bandwidth utilization near the end of the presentation. Imagine a completely symmetric presentation in which the objects have the same number of layers, the same number of bits per layer, and rendering intervals of equal length (including the start up interval). The optimal refined max-min policy will transmit equal layers of each image. However, in addition to these layers, the quality of the last image can be gradually improved during its rendering interval if the client continues to retrieve additional layers.

To see that gradual rendering is applicable for objects other than the first and the last, consider a presentation with three objects, each having 10 layers with 10 K bits per layer. Suppose that 30 K can be transmitted in the start up period prior to the deadline of the first object, 30.01 K in the rendering interval of the first object, and 200 K in the rendering interval of the second object. (The bandwidth available in the rendering interval of the third object is unimportant.) Without gradual rendering, the optimal refined max-min policy will be to send 3 layers of the first object (consuming the 30 K of bandwidth available in the start up interval), 3 layers of the second object (consuming the 30 K of bandwidth available in the rendering interval of the first object), and 10 layers of the third object (consuming 100 K of the remaining available bandwidth). In summary, the optimal refined max-min policy without gradual rendering will be (3, 3, 10).

On the other hand, if we permit gradual 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 gradual 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 gradual rendering, we now consider a policy $P = (j_1, \ldots, j_N)$ to be feasible if the bits sent arrive prior to the end of their rendering intervals. For

illustrative purposes, we consider the following natural definition of quality for gradual rendering.

v_{ij} = rendering time of layer j of object i

w_i = length of the rendering interval for object i

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

Because the calculation of v_{ij} depends on the number of layers chosen for the objects preceding object i, q_i is necessarily a function of these layers and the layers chosen for object i.

The refined max-min algorithm *without* gradual 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 gradual rendering with our proposed heuristic is shown in Fig. 8.

We used the same presentation data in Table 1 of section 6 to compare the layer-oriented refined max-min algorithm with and without gradual rendering. Fig. 9 shows the percentage of layers rendered for each object under the two methods. Note that the policy with gradual rendering is better than the policy without gradual rendering in the sense of the refined max-min criterion. Interestingly, the gradual 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 gradual rendering algorithm delivers some of the layers in images 5 and 10 late. These late arriving layers will contribute less than a full unit towards the quality of the image, 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).

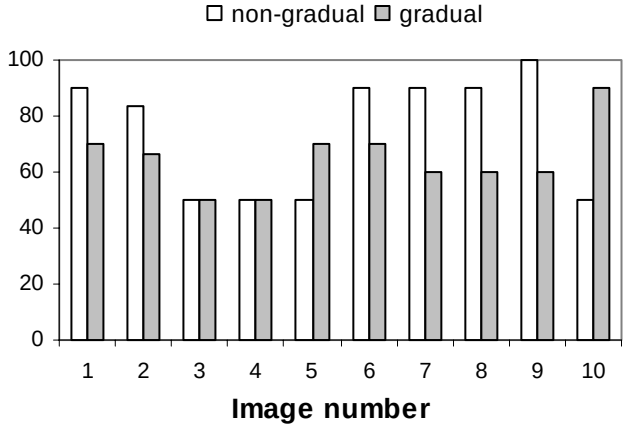

Figure 9: Percentage of layers rendered by the refined max-min algorithm with and without gradual rendering

$S = \{1,\ldots,N\}$

$P = (j_1,\ldots,j_N) = (0,\ldots,0)$

do while $P \neq (L_1,\ldots,L_N)$ and S is nonempty

find $k \in S$ s.t. $q_k(j_k) \leq q_i(j_i)$ for all i in S

  if $(j_1,\ldots,j_k+1,\ldots,j_N)$ is feasible AND

  has better overall quality, then

    $j_k = j_k + 1$

  else

    remove k from S

Figure 8: Refined Max-min algorithm with gradual rendering

8 Related Research

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 [2, 3, 4, 5, 6, 7, 8, 9]. 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. The policy for one-layer static objects described in Section 2 can be viewed as a special case of [2, 4, 6].

Recently, Zhao, Willebeek-LeMair and Tiwari have considered "malleable multimedia presentations" [1, 2]. 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 [2], 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, l, that can be sent while satisfying the bandwidth constraint. In [1] they provide an enhancement algorithm that adds *one layer* to layer l for a subset of the objects while remaining feasible.

Our work differs from [1] and [2] in many respects. First, we allow for general quality values, $q_i(j)$ for each layer j for each object i. References [1] and [2] only consider the special case $q_i(j) = j$, which is in many circumstances not an accurate measure of the quality of a rendered object. Second, 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. For different types of quality value definitions, we develop several efficient algorithms for the refined max-min criterion. We present numerical testing with real progressive JPEG data to investigate and compare the different quality value definitions and optimization criteria. References [1] and [2] do not provide any insight into their model or criteria through numerical work. Finally, we introduce gradual rendering into the optimization objective, and study how gradual rendering influences the optimal policy.

9 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. In order to allow users to tradeoff presentation quality with start-up and interactivity delays, it is desirable to introduce layered encoding into the multimedia presentations. In this paper we have developed a comprehensive methodology for layered multimedia presentations. We have proposed several natural optimization criteria and have developed efficient algorithms for each of the criteria.

Much work remains to be done in the area. From a practical perspective, standards developers need to consider modifying existing document languages for synchronized presentations in order to take into account static object sizes and layers. The current version of SMIL does not provide this information. From a theoretical perspective, optimization criteria with gradual rendering need to be studied in greater detail. We are currently considering all of these issues as well developing client/server implementations of the optimal policies.

References

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

[2] [Zhao 1999] W. Zhao, M. Willebeek-LeMair and P. Tiwari, "Efficient Adaptive Media Scaling and Streaming of Layered Multimedia in Heterogeneous Environments," Proceedings of the IEEE Int. Conference on Multimedia Computing and Systems, Florence, Italy, June 1999.

[2] [Feng 1995] W. Feng and S. Sechrest, "Smoothing and Buffering for the Delivery of Prerecorded Compressed Video," Proceedings of the IS&T/SPIE Multimedia Computing and Networking, February, 1995.

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

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

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

[6] [Salehi 1996] J. Salehi, Z. Zhang, J.F. Kurose and D. Towsley, "Supporting Stored Video: Reduce Rate Variability and End-to-End Resource Requirements through Optimal Smoothing," Proceeding of ACM SIGMETRICS'96, 1996.

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

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

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

[10] Denardo, Eric, *Dynamic Programming.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1982.

[11] Hoschka, P., ed., "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification," Synchronized Multimedia Working Group, W3C Recommendation, 15-June-1998; available at http://www.w3.org/TR/1998/REC-smil-19980615/.

[12] Pennebaker, William B., and Joan L. Mitchell, JPEG Still Image Data Compression Standard, Van Nostrand Reinhold, 1993.

[13] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[14] Independant JPEG Group, JPEG software library, available at http://www.jcu.edu.au/docs/jpeg/.

[15] RealNetworks Web site, http://www.real.com.