Efficient Delivery Techniques for Variable Bit Rate Multimedia*

Yanping Zhao Derek Eager Mary Vernon

Department of Computer Science
University of Saskatchewan
Saskatoon, SK S7N 5A9, Canada
{zhao, eager}@cs.usask.ca

Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706, USA
vernon@cs.wisc.edu

ABSTRACT

Two key technologies enabling scalable on-demand delivery of stored multimedia content are *work-ahead smoothing* and *multicast delivery*. Work-ahead smoothing reduces the burstiness of variable bit rate streams, simplifying server and network resource allocation. Recent multicast delivery techniques such as patching or bandwidth skimming serve clients that request the same content close together in time with (partially) shared multicasts, greatly reducing required server bandwidth.

Although previous studies of work-ahead smoothing have generally assumed very limited client buffer space, in a number of contexts of current interest (such as systems that have significant settop storage), it becomes feasible to fully smooth variable bit rate content. We quantify the start-up delay and settop storage requirements of full smoothing for a number of sample variable bit rate objects. We then evaluate a fundamental conflict between aggressive smoothing and the new multicast delivery techniques. Work-ahead smoothing requires sending data for high rate portions of an object earlier than it is needed for playback, while multicast techniques yield their greatest benefit when data is delivered within each stream as late as possible so that more clients can share reception of that data. A new multicast delivery technique is proposed that can accommodate aggressive smoothing with increased efficiency in comparison to previous techniques, particularly for high request rates.

Keywords: streaming media, multicast, bandwidth skimming, variable bit rate, work-ahead smoothing

1. INTRODUCTION

A number of emerging applications require efficient methods for on-demand streaming of popular stored multimedia content, such as movies, news clips, or lecture videos. Generally, the straightforward approach of delivering a separate stream of uncompressed video to each client will consume too much server, network, and client bandwidth to be feasible.

Compression[15] can reduce the bandwidth requirements of streaming video by one to two orders of magnitude. Compression yields either *variable quality* content, in which quality is degraded during scene changes and periods of high motion or greater detail, or *variable bit rate* (VBR) content. Constant-quality, VBR video has been shown to exhibit substantial rate variability, on time scales as long as several minutes[11]. Such rate variability greatly complicates the task of designing server and network resource management mechanisms that can efficiently support jitter-free playback.

The problem of effectively delivering VBR content can be addressed by *work-ahead smoothing*[21]. Work-ahead smoothing reduces rate variability by streaming data from high rate portions of an object before it is needed, during otherwise low rate periods, and buffering it at the client until playback. The extent to which smoothing can take place is constrained by the available client buffer space. Although most prior work on work-ahead smoothing has assumed that clients have very limited buffer space (i.e., a few megabytes or less), in many contexts of current interest (such as delivery to PCs or to settops containing disk storage) clients have sufficient buffer space to permit *full smoothing* to a constant bit rate (CBR) stream. The start-up latency and

* This work was supported by the Natural Sciences and Engineering Research Council of Canada under Grant OGP-0000264 and by the National Science Foundation under Grant CCR 9975044.

storage space required for full smoothing of several MPEG-1 video traces are quantified and shown to be quite modest in Table 1 of Section 3.

A second key technology that enables efficient on-demand streaming of stored multimedia content is multicast (or broadcast) delivery techniques that are able to serve many clients requesting the same content at different times with only a few streams. Of particular interest in this work are *patching*[2, 3, 9, 13], *hierarchical stream merging*[5, 6], and *bandwidth skimming*[7, 16]. These delivery techniques require no *a priori* division of objects into "hot" and "cold" sets, and allow the server to start streaming an object immediately upon receiving a request for it. These techniques enable a client to "catch up" to an earlier client being served the same content, at the cost of requiring the newer client to receive data at a higher rate, and some additional client storage space for buffering data that is received ahead of when it is needed. Once a client has caught up to another client or group of clients, they can share a single multicast stream.

The questions addressed in this paper are motivated by the observation that there is a fundamental conflict between aggressive work-ahead smoothing and scalable on-demand streaming. That is, providing on-demand streaming requires frequent transmissions of the earlier portions of a media object, while providing scalable delivery requires highly shared and relatively infrequent transmissions of the latter portions of an object. Since work-ahead smoothing involves moving data from high rate portions of an object to otherwise low rate portions closer to the beginning, smoothing has the side-effect in the context of scalable on-demand streaming of increasing the frequency with which this data must be delivered. To understand the impact of this conflict on the design of scalable streaming protocols, we address the following questions:

- How much rate variability occurs in existing VBR videos as well as in existing composite objects, where a composite object is a collection of synchronized components (e.g., an audio clip together with a sequence of static images)?
- What is the minimum server bandwidth required to deliver media objects that have rate variabilities that occur in practice, and is this server bandwidth significantly lower than with the straightforward approach of applying an existing scalable delivery protocol to an aggressively smoothed stream?
- If there is significant room for improvement over the straightforward approach, can we devise a practical new scalable delivery protocol that achieves significantly more efficient delivery of popular VBR objects?

The principal contributions of the paper are (1) quantification of the rate variability in existing VBR objects, (2) a simple equation for computing the lower bound on required server bandwidth for delivering a given VBR object under a specified maximum client start-up delay, and (3) a new delivery protocol, VBRBS, that has fixed-rate server streams, yet exploits knowledge of the VBR profile to more effectively aggregate clients and conserve server bandwidth. The new delivery technique is derived from the bandwidth skimming protocol. A similar approach could be applied to other techniques such as patching.

The remainder of the paper is organized as follows. Section 2 provides background on work-ahead smoothing and on bandwidth skimming. Section 3 evaluates the start-up delay and storage requirements of full smoothing for a variety of VBR video traces, composite multimedia objects, and synthetic objects. The impact of rate variability and of full smoothing on the inherent server bandwidth requirements for on-demand streaming of stored content is studied in Section 4. Section 5 presents the new *VBR Bandwidth Skimming* delivery technique, and presents simulation results assessing its performance. Variants of this technique that support delivery to clients with limited storage capacity are described and evaluated in Section 6. Section 7 concludes the paper.

2. BACKGROUND

2.1. Work-ahead Smoothing

Streams with high bit rate variability greatly complicate the task of allocating server and network resources. Basing resource allocation and provisioning on peak rates is inherently inefficient, as peak rates may be more than an order of magnitude higher than average rates. Furthermore, the achievable transmission rate to a particular client may be sufficient to accommodate the average rate of a VBR stream, but not the peak rate.

If the achievable transmission rate to a client is high enough for the peak rate of unsmoothed VBR content, it would seem preferable, from the client perspective at least, to receive instead higher quality (but smoothed) content. For these reasons, it is highly desirable to reduce the rate variability of individual streams, a task that is accomplished by work-ahead smoothing.

A number of work-ahead smoothing techniques have been proposed in the literature. These techniques differ in how client buffer size constraints impact various properties of interest (peak rate, rate variability, number of rate changes, as well as others)[8]. This paper focuses for the most part on the case in which clients have sufficient available storage space, for example a commodity disk, to permit full smoothing of VBR content to a CBR stream. Full smoothing minimizes the peak rate, thus permitting delivery of the highest possible quality content as constrained by the achievable transmission rate to the client, as well as enabling the most efficient server and network resource allocation.

Section 6 investigates variants of VBR bandwidth skimming for clients with more limited storage capacity. In this case, the variant that delivers the "smoothest" streams uses the *optimal smoothing* algorithm of Salehi et al.[21]. This algorithm minimizes peak rate and rate variability, subject to specified constraints on the transmission schedule.

2.2. Bandwidth Skimming

Of interest in this paper are the multicast delivery techniques (including patching[2, 3, 9, 13], hierarchical stream merging[5, 6], and bandwidth skimming[7, 16]) that operate over low and high request arrival rates, and allow the server to start streaming an object to a client immediately upon receiving the client request. To date, these techniques have been developed for constant bit rate streaming only. This is in contrast to the *segmented, periodic broadcast* techniques in which objects are divided into segments that are continuously broadcast, independent of client requests[12, 14, 16–19, 22, 24]. Use of these latter techniques with VBR objects has been considered in previous work[12, 17, 18, 22]; a common approach[12, 17, 18] is to fully smooth each segment, and construct the transmission schedule so that each segment can be completely received by clients prior to its play point.

Bandwidth skimming is distinguished by its low required aggregate transmission rate to each client. Other than bandwidth skimming and the *Optimized PB* protocol[16], multicast techniques for on-demand streaming require that the achievable transmission rate to each client be at least twice the object bit rate, so that clients can "catch up" to other clients that have made earlier requests for the same object by concurrently receiving two or more full-rate (or equivalent) transmission streams. In contrast, bandwidth skimming uses only a small "skim" of the achievable transmission rate to support this dynamic client aggregation, allowing most of the bandwidth to the client to be consumed in supporting streaming of the highest possible quality content. This objective is achieved through use of the hierarchical stream merging approach to aggregating clients, together with a mechanism for using the bandwidth skim to effect each aggregation.

An example of hierarchical stream merging is depicted in Figure 1 for the simple case in which the achievable transmission rate to each client is assumed equal to at least twice the (constant) object bit rate, and thus clients can receive two full-rate streams simultaneously. In the scenario depicted in the figure, four clients request the same object and are each initially provided with their own multicast stream. Clients B and D also begin listening to the streams that were initiated for clients A and C, respectively, thus accumulating data at twice the object bit rate, as indicated in the figure by the dashed lines. At times $T2$ and $T5$, respectively, clients B and D have accumulated sufficient data that their own streams can be terminated as long as they continue listening to the streams initiated for clients A and C, respectively. After clients C and D have been aggregated in this manner, both begin listening to the stream initiated for client A, thus accumulating data at twice the object bit rate and allowing the stream created for client C (and now received by both clients C and D) to be terminated at time $T6$.

Variants of hierarchical stream merging differ according to the precise policy used to determine which clients to aggregate with which others, and in what order, as well as according to what streams are listened to by clients so as to accomplish the desired aggregations. For the results presented in this paper, we have adopted the *Earliest Reachable Merge Target* (ERMT) policy[6], in which each client (or group of aggregated clients) tries to "catch up" to the closest earlier client (or group of clients) that is "catchable", if any. The target client may

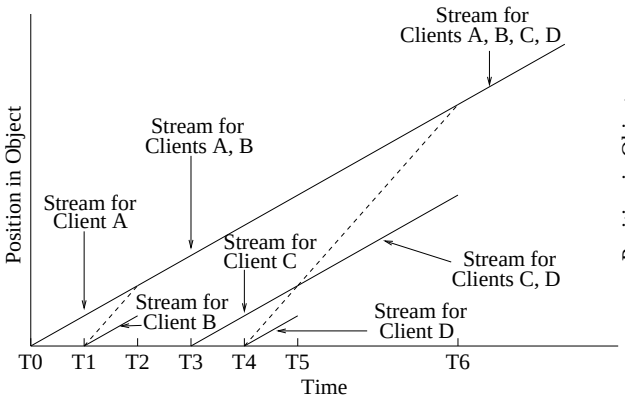


Figure 1: Hierarchical Stream Merging

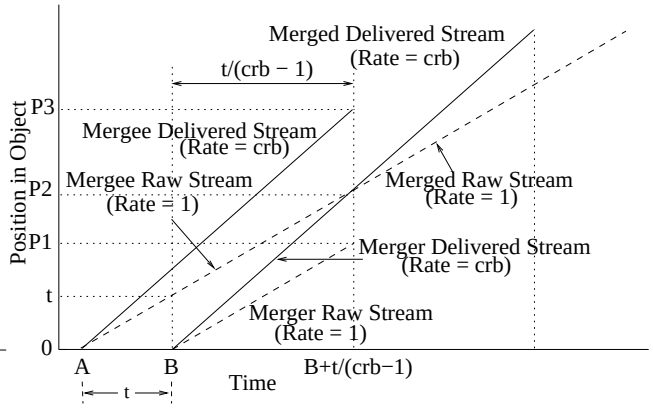


Figure 2: Latest Patch

be simultaneously trying to catch some other client, with the successful merge being the one that occurs first. There are a number of other policies[1, 4, 6] that would give similar results.

When only a small "skim" of the bandwidth to the client is available to support client aggregation, the mechanism of listening on two full-rate streams so as to catch up to earlier clients cannot be used, and a new mechanism is needed. The two most promising previously defined bandwidth skimming policies are *Partition* and *Latest Patch*[7]. Partition was shown to have highest performance among the previously studied bandwidth skimming policies, whereas Latest Patch has reasonably good performance and is simpler to implement. Each uses hierarchical stream merging together with a different mechanism for using the skim bandwidth to aggregate clients. Latest Patch is illustrated in Figure 2 and described below.

The Latest Patch policy is defined for an arbitrary achievable transmission rate to the client, or "client receive bandwidth" (*crb*), greater than the (constant) object bit rate. In Figure 2 the object bit rate is chosen as a unit of measurement; i.e., fixed at 1. The data required for a merger client to catch the mergee is delivered by a merger stream at rate *crb*, as illustrated in the figure. This figure supposes that the mergee is itself attempting to catch an earlier client, and distinguishes between the actual streams that run at rate equal to the *crb*, and the rate 1 "raw" streams that are not delivered by the server, but simply define the rate at which the data is being viewed or played by the client. (For aggregated clients, the raw stream is defined according to the viewing/playing point of the client that is furthest along in the object.) The implementation approach for Latest Patch that is illustrated in Figure 2 forms the basis of the VBR bandwidth skimming technique presented in Section 5, and, in fact, the terminology and drawing of Figure 2 anticipates this technique.

3. *VBR* OBJECT CHARACTERISTICS

The analyses and simulations in this paper use traces from a variety of VBR videos, as well as bit rate profiles from two "composite" objects consisting of a mixture of media types. Synthetic bit rate profiles are also used, as these yield additional insight into how rate variability impacts server bandwidth requirements and the relative performance of various delivery techniques. The characteristics of these objects are summarized in Table 1.

The video traces are of 16 MPEG-1 encoded video segments[20] and an MPEG-1 encoding of the movie *starwars*[11]. The video segments were created using the UC Berkeley MPEG-1 software encoder. Each contains 40,000 frames, representing 26.7 minutes of video at 25 frames per second. The encoder input was 384×288 pels with 12 bit color information. The 121 minute *starwars* video consists of 171,000 frames with a frame rate of 24 frames per second (i.e., the original film rate). The original video was captured as 408×508 pels, and then interpolated and filtered to standard CIF frame size, which is 240×352 (Luminance - Y) and 120×176 (Crominance - U & V). All videos use the sequence of MPEG I, P and B frames "IBBPBBPBBPBB". For each video, Table 1 gives the average bit rate, the average, maximum, minimum, and standard deviation of the frame sizes, the start-up delay and the client-side buffering requirements assuming a CBR transmission at rate equal to the average object bit rate or 1.2 times the average bit rate, and the delivery bit rate (relative to the

Table 1: Object Characteristics

Video Name	Avg Rate (Mbps)	Frame Sizes (KBytes)				Start-up Delay (%)		Storage (%)		Rate/Avg (Delay=0)
		Avg	Max	Min	Std	Rate=Avg	1.2Avg	Rate=Avg	1.2Avg	
asterix	0.56	2.79	18.4	0.04	2.52	0.43	0.00	6.23	19.2	1.02
dinosaur	0.33	1.63	15.0	0.11	1.84	6.62	0.04	7.26	12.4	1.21
fuss	0.68	3.39	23.4	0.31	3.25	0.91	0.00	2.74	16.7	1.07
lambs	0.18	0.91	16.8	0.04	1.40	4.09	0.66	10.0	16.7	3.83
movie2	0.36	1.79	21.6	0.03	2.36	3.85	0.20	5.26	17.5	1.26
mrbean	0.44	2.21	28.6	0.04	2.58	3.45	0.74	9.45	21.3	2.73
mtv1	0.62	3.08	28.7	0.05	2.88	4.87	0.03	6.34	13.5	1.81
mtv2	0.49	2.47	31.4	0.06	2.68	6.31	0.02	11.8	21.2	1.29
news2	0.38	1.92	23.7	0.03	2.44	1.96	0.01	4.68	16.4	1.42
race	0.77	3.84	25.3	0.52	2.65	1.83	0.07	2.90	17.1	1.38
simpsons	0.46	2.32	30.1	0.04	2.58	3.98	1.50	3.98	15.6	1.90
soccer	0.63	3.14	23.8	0.37	2.66	1.33	0.16	4.85	16.9	1.93
superbowl	0.59	2.94	17.6	0.04	2.34	1.38	0.00	4.27	15.8	1.06
talkshow1	0.36	1.82	13.4	0.26	2.06	2.44	1.16	4.02	18.3	2.58
talkshow2	0.45	2.24	16.6	0.45	2.28	1.99	0.01	4.27	18.5	1.52
terminator	0.27	1.36	9.95	0.04	1.27	2.02	0.25	2.97	15.5	1.30
starwars	0.37	1.95	23.2	0.06	2.27	0.51	0.00	6.58	21.1	1.18

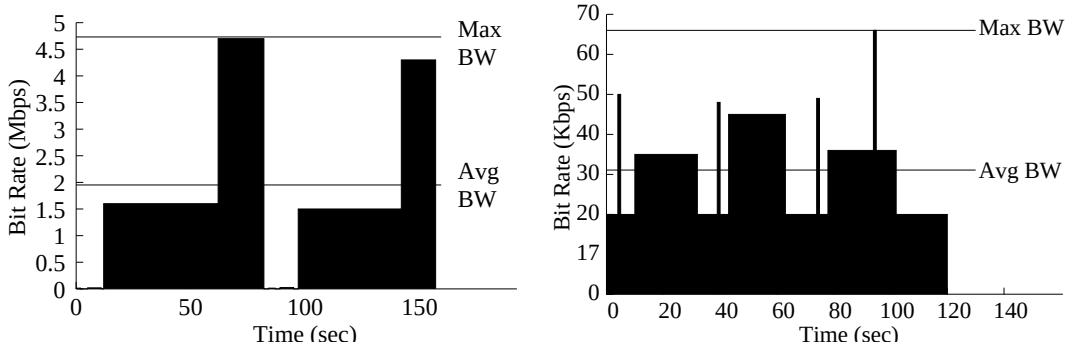
Name	Avg Rate (Mbps)	Peak Rate (Mbps)	Start-up Delay (%)		Storage (%)		Rate/Avg (Delay=0)
			Rate=Avg	1.2Avg	Rate=Avg	1.2Avg	
composite 1	1.95	4.70	4.62	0.00	18.0	26.5	1.09
composite 2	0.03	0.07	5.35	0.00	7.73	12.2	1.10
synthetic 1	0.55	1.0	40.9	25.8	40.9	30.9	1.82
synthetic 2	0.55	1.0	0.00	0.00	40.9	50.9	1.00

average object bit rate) required for no start-up delay. Here *start-up delay* is defined to be the time from when the client begins receiving the stream until the client can begin to view/play the object, neglecting any delay component required to hide network jitter.

The required start-up delay depends on the relationship between the initial bit rate of the video (as determined by the sizes of the initial frames) and the initial streaming rate. The amount of client-side storage required for buffering video data depends on the object's rate variability, and the degree and type of work-ahead smoothing. Table 1 shows start-up delays expressed as a percentage of the object playback duration and storage requirements expressed as a percentage of the object size, for the cases of full smoothing to a CBR stream at the average object bit rate ("Rate=Avg"), and full smoothing to a CBR stream at rate equal to 1.2 times the average bit rate. Note that when the streaming rate is equal to the average bit rate, an appreciable start-up delay is required in some cases. Delivery at 1.2 times the average bit rate largely eliminates this start-up delay, but at the cost of increasing the amount of required buffering at the client.

A *composite* multimedia object consists of a number of components of potentially differing types, such as static images, text files, video clips, and/or audio clips, organized into a coherent multimedia presentation with specified relative viewing/playing times[10, 23]. Considered here are bit rate profiles from two composite objects[23]. *Composite object 1* is a 157 second report of an Olympic swimming competition, and includes a number of images, narrations, and video clips. *Composite object 2* provides a two minute guided tour of Washington D.C., and consists of a sequence of images with accompanying narrations, together with background music. Figure 3 gives the unsmoothed bit rate profiles of these objects, while Table 1 provides summary statistics.

The bandwidth requirement with multicast delivery and the operation of work-ahead smoothing are greatly impacted by the bit rate at the beginning of an object. The two synthetic bit rate profiles that are considered



(a) Composite Object 1 (b) Composite Object 2

Figure 3: Composite Object Bit Rate Profiles

here represent opposite extremes. Each profile consists of two CBR sections of equal duration. In *synthetic 1*, the initial section has high bit rate while the remainder has low bit rate; in *synthetic 2*, the positions of these two sections are reversed. As shown in Table 1, both profiles have the same average and peak bit rate, and both have relatively high client storage requirements when smoothed to a CBR stream. Owing to its high initial bit rate, however, *synthetic 1* requires a large start-up delay, while *synthetic 2* requires no start-up delay.

4. IMPACT OF VBR ON BANDWIDTH NEEDED FOR ON-DEMAND DELIVERY

This section addresses the following two basic questions:

- How does bit rate variability impact the server bandwidth required for scalable on-demand streaming using multicast delivery techniques?
- What is the impact on server bandwidth requirements of smoothing a variable bit rate object and delivering it as if it were a constant bit rate object?

These questions are addressed through consideration of fundamental bounds instead of within the context of a single delivery technique, so as to obtain more broadly applicable insight.

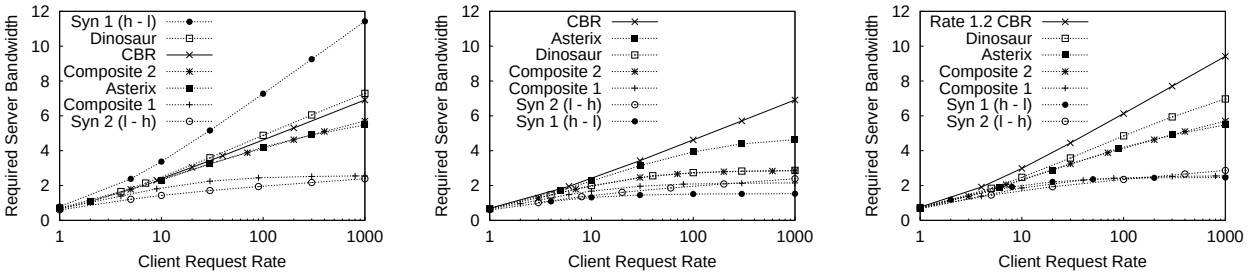
A tight lower bound $B^{CBR}_{minimum}$ on the required server bandwidth† for *any* delivery technique providing immediate on-demand streaming of a constant bit rate object, with no start-up delay, is as follows[5]:

$$B^{CBR}_{minimum} = \int_0^T \frac{dx}{x + \frac{1}{\lambda}} = \ln(T\lambda + 1) = \ln(N + 1), \tag{1}$$

where server bandwidth is in units of the (constant) object bit rate, T is the playback duration of the object, λ is the average object request rate, and $N = \lambda T$ is the average number of requests for the object per period of length T. The above bound is derived by considering a small portion of the object at some arbitrary time offset x. For an arbitrary client request that arrives at time t, this portion of the object can be delivered no later than time $t + x$, in order that the client can begin to view/play the object immediately and without jitter. If this portion is multicast at time $t + x$, then (at best) those clients that request the file between time t and $t + x$, could receive this multicast. If the average time from $t + x$ until the next request for the object is $1/\lambda$, then the minimum frequency of multicasts of the portion at time offset x is $1/(x + 1/\lambda)$, yielding the above bound.

The bound given by equation (1) assumes Poisson request arrivals, and no constraint on the aggregate rate of transmissions that clients can receive concurrently. The former assumption can be relaxed to yield a more general but very similar analytic result[5]. The latter assumption as well does not appear to impact the insights obtained here, as will be shown in Section 5.

†The "required server bandwidth" for an object and delivery technique is defined as the average server bandwidth the delivery technique uses to satisfy client requests for that object.



(a) No Start-up Delay

(b) Start-up Delay Equal to that Required for CBR Delivery at the Average Bit Rate

(c) Start-up Delay Equal to that Required for CBR Delivery at 1.2 x the Average Bit Rate

Figure 4: Lower Bounds on Required Server Bandwidth for Delivery of VBR Objects

The two questions raised at the beginning of this section are addressed through an extension of the above analysis to the delivery of a VBR object. Without loss of generality, define a VBR object as a sequence of m CBR segments $<< t_0, b_0 >, < t_1, b_1 >, ..., < t_{m-1}, b_{m-1} >>$, where the i^{th} segment starts at time offset t_i from the beginning of the object ($t_0 = 0$; $0 < t_i < T$ for $i > 0$) and has bit rate b_i ($b_i \geq 0$) measured in units of the average bit rate for the object. A tight lower bound $B^{VBR}_{minimum}$ on the required server bandwidth for on-demand streaming, with no start-up delay, can be derived as:

$$B^{VBR}_{minimum} = b_0 \int_{t_0}^{t_1} \frac{dx}{x + \frac{1}{\lambda}} + b_1 \int_{t_1}^{t_2} \frac{dx}{x + \frac{1}{\lambda}} + ... + b_{m-1} \int_{t_{m-1}}^{T} \frac{dx}{x + \frac{1}{\lambda}} \quad (2)$$

$$= b_0 \ln(\frac{t_1\lambda + 1}{t_0\lambda + 1}) + b_1 \ln(\frac{t_2\lambda + 1}{t_1\lambda + 1}) + ... + b_{m-1} \ln(\frac{T\lambda + 1}{t_{m-1}\lambda + 1}).$$

The above bound can also be applied for a non-zero start-up delay, by adding a CBR segment at the beginning of the object of length equal to the start-up delay, and bit rate of zero.

Figure 4 provides lower bounds on required server bandwidth for six of the VBR objects described in Section 3. Results for the other VBR objects are qualitatively similar. The x-axis is the normalized client request rate for the object, N, and the y-axis is the required server bandwidth in units of the object's average bit rate.

In Figure 4(a) the CBR curve shows the lower bound on required server bandwidth, computed using equation (1), for a true CBR object, which has no start-up delay when delivered at the object's average bit rate. The bound for each VBR object is computed using equation (2) with no start-up delay. Note that bit rate variability has a large impact on the minimum required server bandwidth. Particularly under high request rates, the minimum required server bandwidth is largely determined by the bit rate of the beginning of the object, since this portion must be delivered most frequently. This point is illustrated clearly by the two synthetic objects.

In Figure 4(b) the CBR curve gives the minimum required server bandwidth, as computed using equation (1), when a VBR object is fully smoothed and delivered at the average object bit rate, as if it were a CBR object. Note that this is the same required server bandwidth as for a true CBR object (Figure 4(a)), but a start-up delay is implied for the VBR object. For each VBR object, the graph shows the minimum required server bandwidth computed using equation (2) with an additional term for the start-up delay (given in the "Rate=Avg" column of Table 1) that would be required for CBR delivery of the fully smoothed object. Note that equation (2) assumes that after the start-up delay, each segment of the VBR object is delivered as late as possible and at the bit-rate that the segment is consumed. These curves show that there is a substantial increase in the minimum required server bandwidth, particularly at high request rates, when a VBR object is fully smoothed and delivered as a CBR object as compared with delivering the unsmoothed object at the variable bit rate.

Figure 4(c) gives the minimum required server bandwidth when the VBR object is fully smoothed and delivered at 1.2 times the average bit-rate (thus reducing or eliminating the start-up delay, as shown in the

"1.2Avg" column of Table 1), and for each of the six VBR objects assuming the object was not smoothed and was delivered at its variable bit rate after the lower client start-up delay. This figure shows a similar performance penalty as in Figure 4(b) for creating a CBR object from the VBR object.

5. VBR BANDWIDTH SKIMMING

The results of the previous section indicate that fully smoothing a VBR stream and then applying bandwidth skimming (or another multicast stream merging technique) to the CBR stream requires more server bandwidth than necessary for many compressed videos and composite objects and most target start-up delays. Thus, this section develops a new multicast delivery technique, called *VBR Bandwidth Skimming* (VBRBS), that exploits the variable bit rate profile of the object being delivered. It is assumed here that the available client storage space permits full smoothing to a constant bit rate stream at the achievable transmission rate to the client (*crb*).[‡] Section 6 describes variants of the technique that support delivery to clients with more constrained buffering capability.

VBRBS is based upon the implementation of the Latest Patch delivery technique illustrated in Figure 2, with the following four key changes, illustrated in Figure 5:

- The "raw" streams shown in Figure 5 are now variable bit rate streams, corresponding to play/view progress through a VBR object. The "delivered" streams now serve two purposes: catching up to an earlier client (as before), and also work-ahead smoothing.
- In VBRBS the delivered stream is *always* a constant bit rate stream at rate *crb*, even if there is no earlier catchable client. Since all of the streams transmitted by the server are constant bit rate, server and network resource allocation may be greatly simplified.
- A "merge point" (i.e., a point at which two clients or groups of clients can be aggregated) can no longer be simply defined by the intersection of a delivered stream and a raw stream (as illustrated in Figure 5); instead, merge points are defined based on when a delivered stream *dominates* a raw stream, in the sense that it will achieve the same or later position in the object, in comparison to the variable bit rate raw stream, at all subsequent points in time. More precisely, a merge point should satisfy the following constraints: (1) the merger's delivered stream should have achieved the same position in the object as the mergee's raw stream, (2) at any time t after the merge point until the end of delivery, the merger's delivered stream (i.e., the delivered stream of the aggregated clients) should have achieved the same or later position in the object as the mergee's raw stream (i.e., the raw stream of the aggregated clients), and (3) since we wish clients to be aggregated as quickly as possible, the merge point is the *earliest* point satisfying the first two constraints. An efficient off-line analysis that can be used for determining merge points is described below.
- For a dynamic merging policy such as ERMT or CT[6], VBRBS employs a variant of the policy that can be viewed as a hybrid between dynamic and "static pairing" approaches. The new policy variant is motivated by the potential bandwidth cost of delivering all streams at rate *crb*. Note that whenever a client is caught by a later client, any object data that has been delivered from beyond the client's view/play point when caught is "wasted" in the sense that the server will be delivering it again for the merger client. Although perhaps surprisingly, our results indicate that the performance cost of this wasted bandwidth is often not excessive, the cost becomes more substantial as *crb* increases. To address this problem, the VBRBS merging policy commits to aggregating two groups of clients substantially ahead of when the aggregation can actually occur (in the results presented here, after a time duration equal to only a third of the time required for the merger delivered stream to reach the merge point, from its previous merge point if the merger delivered stream has participated in an earlier merge, or from the beginning of the stream if not). Once an aggregation is committed to, the delivered stream for the mergee can be shut off before the aggregation takes place, specifically, after it has reached the position in the object defined by the merge point. Although potentially this new policy might lead to poorer choices of which clients to aggregate, in our simulations this has not been found to have a significant performance impact.

[‡]Client heterogeneity can be addressed with standard techniques such as layered encoding or multiple object versions.

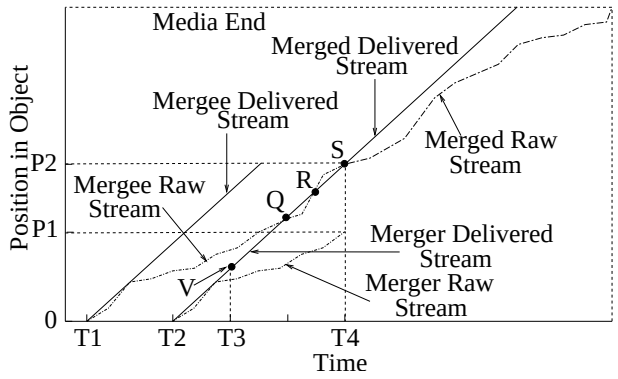


Figure 5: VBR Bandwidth Skimming

```
Procedure Generate_dominating_table(d)
1.   /* d = delivered stream rate
2.   i ← M     /* M = number of frames
3.   while (i > 0) {
4.      while (b_i < d) {     /* b_i = bit rate of frame i
5.         /* R = frame rate
6.         /* S_i = total # of bits in first i frames
7.         add [i/R − S_i/d, i] to table
8.         i ← i − 1 }
9.      j ← i − 1
10.     /* F_i = frame i
11.     while((slope(F_j, F_i) > d) AND (j > 0))
12.        j ← j − 1
13.     i ← j }
```

Figure 6. Algorithm for Generating a Dominating Table

Figure 5 illustrates client aggregation in VBRBS. The mergee client's delivered stream and its raw stream indicating its play/view progress start at time $T1$, while the merger client's streams begin at time $T2$. The merger client's delivered stream and the mergee client's raw stream intersect at the points Q, R, and S. Each of these points satisfy the first of the constraints on merge points defined above. However, between the point R and the point S, the mergee's raw stream reaches later positions in the object than the merger's delivered stream. Therefore, the points Q and R violate the second constraint. The point S is the earliest point satisfying both constraints, and thus it defines the merge point. Note that with the "early commit" used in VBRBS, the merge is committed to at time $T3$, at point V in the merger delivered stream, and the mergee's delivered stream can therefore be shut down once it has delivered data up to position $P2$. Without early commit, in contrast, the stream would continue until time $T4$, even though the data delivered from $P2$ on would be "wasted" as this data must be delivered again for the merger client. In comparison to Latest Patch, VBRBS is more efficient in two main respects: (1) the merger stream can catch the mergee stream sooner than if the mergee stream was smoothed, and (2) the mergee stream is terminated even *before* the merge occurs.

One approach to determining when two clients or groups of clients can be aggregated (i.e., determining merge points) entails computing off-line a "dominating table" for each VBR object. For example, consider a VBR video, in which case each entry in the dominating table contains two fields: a *separation time* t and a *dominating frame number* i, indicating that if the time separation between the merger and the mergee clients is equal to the time t, then the merger and the mergee can be aggregated when the mergee's raw stream reaches the i^{th} frame of the video. Using the dominating table, the merge time for mergee and merger clients with request times t_e and t_r, respectively, can be determined from the entry with the *smallest* separation time t that is greater than or equal to $t_r - t_e$. The merge time is computed as $t_e + i/R$, where i is the dominating frame number, and R is the frame rate.

Figure 6 outlines an $O(M)$ off-line algorithm for generating the dominating table for a VBR video, and defines notation d, M, b_i, R, S_i, and F_i. Although the algorithm as shown assumes a granularity of frames and does not restrict the number of table entries, it is straightforward to modify the algorithm so that it only adds an entry whose separation time differs by some minimum value from that of the previously added entry, and therefore bounding the space usage to a reasonable value. The algorithm loops through frames from last to first, with two main cases depending on the comparison between b_i and d for a frame i:

- *Case 1*: $b_i < d$ (*Lines 4 – 8*). In this case, as illustrated in Figure 7 (a), the current frame i is a potential merge point. The corresponding separation time can be computed as $t = \frac{i}{R} - \frac{S_i}{d}$, where $\frac{i}{R}$ is the time duration from the raw stream's starting time to the merge point, and $\frac{S_i}{d}$ is the time duration from the delivered stream's starting time to the merge point.

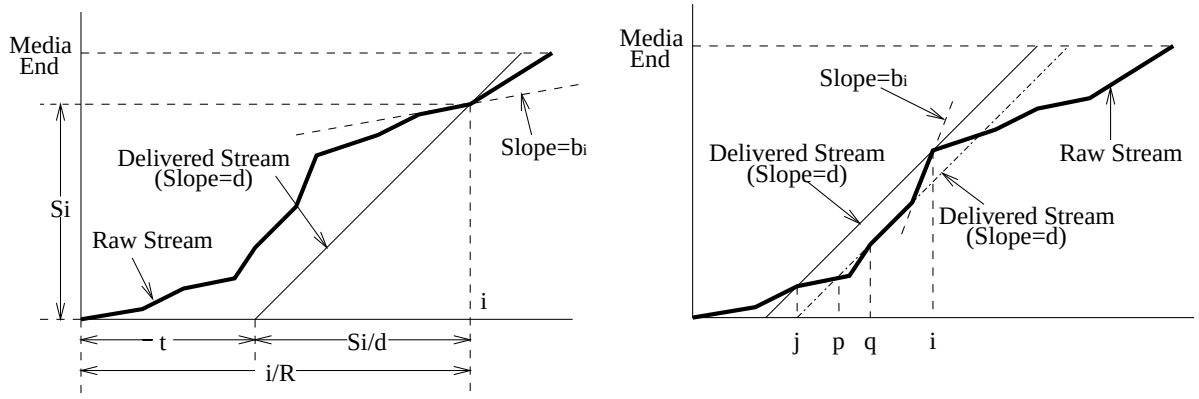
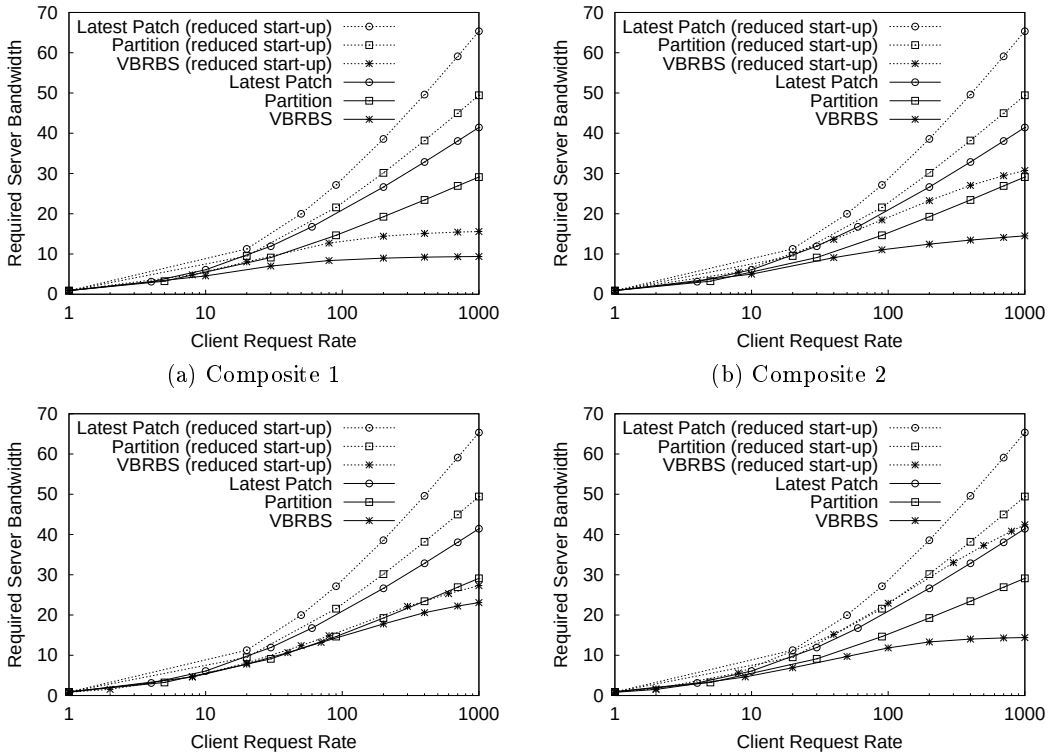


Figure 7: Two Scenarios in the Algorithm of Figure 6



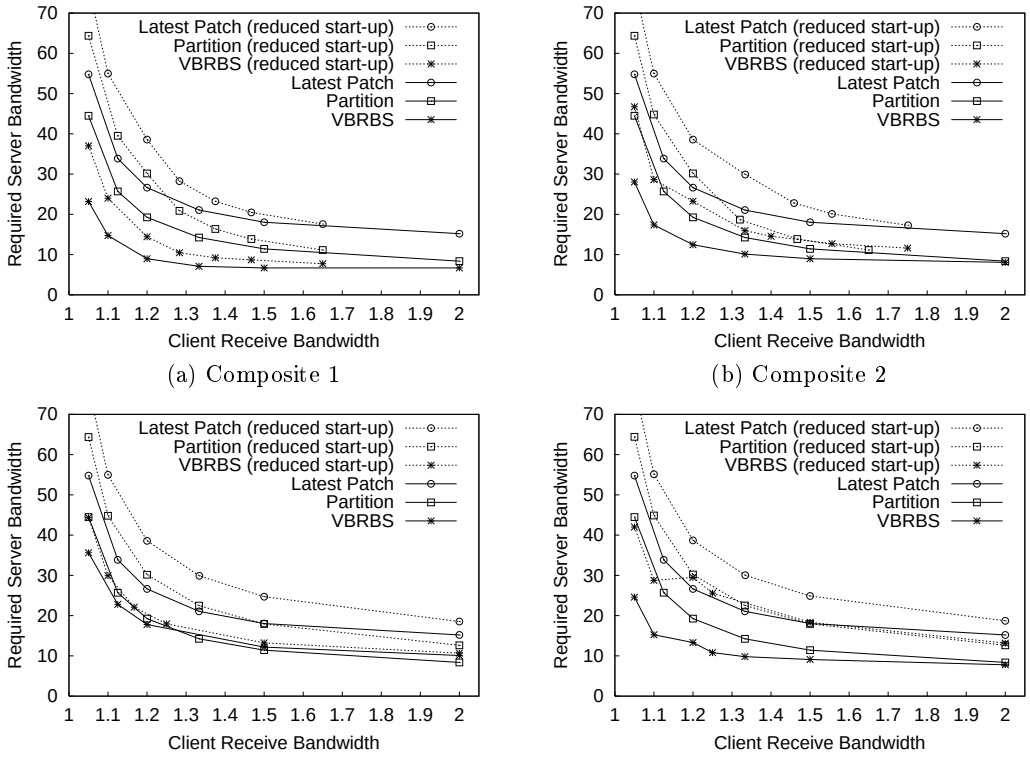
(a) Composite 1 (b) Composite 2

(c) Asterix (d) Dinosaur

Figure 8: Performance of VBRBS, Partition, and Latest Patch versus Client Request Rate ($crb = 1.2$)

- *Case 2*: $b_i \geq d$ (*Lines 9 – 13*). This case is illustrated in Figure 7 (b). Assuming that the next point where an intersecting delivered stream would intersect with the raw stream, if such a point exists, is at frame j, then any frame from $j+1$ to i (e.g., frames p and q) can not be potential merge points. Therefore, the next frame to check is frame j.

Figures 8 and 9 compare the required server bandwidth for delivery of a VBR object using VBRBS, to that required if the object is fully smoothed and then delivered as a constant bit rate object using the Partition



(a) Composite 1 (b) Composite 2

(c) Asterix (d) Dinosaur

Figure 9: Performance of VBRBS, Partition, and Latest Patch versus crb ($N = 200$)

or Latest Patch techniques[7]. Results are presented for the two composite objects, and for two of the videos (*asterix* and *dinosaur*), described in Section 3. Similar results are obtained for the other objects. The y-axis in each graph is the required server bandwidth in units of the object's average bit rate. In Figure 8, the x-axis is the normalized client request arrival rate N. The value of crb is fixed at 1.2 (in units of the object's average bit rate). In Figure 9, the x-axis is crb, and the normalized client request arrival rate is fixed at 200. These results were generated using simulations in which client request arrivals are Poisson; qualitatively very similar results were obtained using a heavy-tailed distribution of interrequest times modeled by a Pareto distribution.

Each graph in Figures 8 and 9 includes six curves, two for each technique. For the curves labelled Partition and Latest Patch, these techniques are applied to a VBR object that has been fully smoothed to a constant bit rate equal to the average object bit rate. Start-up delay is required in this case, with values as indicated in Table 1. For the curve labelled VBRBS, this technique is applied with the same start-up delay added to the "raw" stream defining play/view progress. For the curves with the labelling Partition or Latest Patch "reduced start-up", the respective technique is applied to the object after it has been fully smoothed to a rate equal to the minimum of: (a) the rate needed to reduce the start-up delay to zero, and (b) the average object bit rate plus one-half the amount by which crb exceeds the average object bit rate (thus achieving a compromise between using extra bandwidth for stream merging, and using it to reduce start-up delay).[§] Whatever start-up delay is then required (if any), is added to the raw stream used by the VBRBS technique, for the curve labelled VBRBS "reduced start-up". Note that reduced start-up delay generally implies higher required server bandwidth. (These variations in start-up delays explain the occasional raggedness in the VBRBS curves.)

[§]For Partition, it is also necessary to consider only points such that crb is equal to $1 + 1/m$ times the smoothed object bit rate, for some positive integer m.

The key observations from these figures are:

- All of the considered techniques are scalable, in that required server bandwidth for high request rate grows only logarithmically.
- The required server bandwidth with VBRBS is substantially lower than that with Latest Patch.
- For low client bandwidth "skims" (i.e., *crb* less than 1.5), and high client arrival rates (i.e., $N \geq 100$), the required server bandwidth with VBRBS is (sometimes substantially) lower than that with Partition. In other cases these two policies have similar required server bandwidth.
- The extent of improvement provided by VBRBS depends on how much variability there is in the bit rate profile of the object, particularly the beginning portion. The analysis of Section 4 yields useful insight in this respect. For example, Figure 4(b) and (c) indicate less scope for reducing required server bandwidth with *asterix* with smoothing to the average object bit rate, and *dinosaur* with smoothing to 1.2 times the average object bit rate, than in the other cases considered in the figure, which is consistent with the results in Figures 8 and 9.

6. ACCOMMODATING CLIENTS WITH LIMITED STORAGE CAPACITY

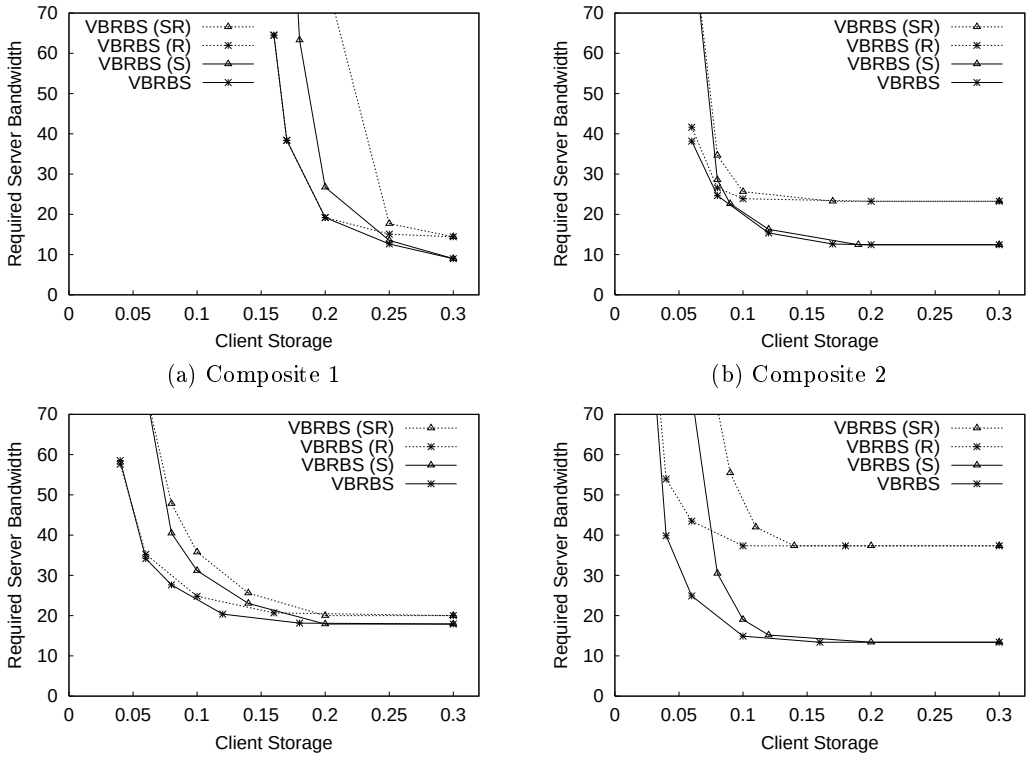
Recall from Table 1 that the client buffer space requirements for VBRBS with *crb* equal to 1.2 times the average object bit rate are approximately 20-25% of the object, or less, for the MPEG-1 video or composite objects. This section considers two variants of VBRBS for the case that clients have insufficient buffer capacity to receive the entire object at fixed rate equal to *crb*. These variants adopt the same basic approach illustrated in Figure 5, but with delivered streams that are not constant bit rate.

The two variants make differing tradeoffs between the smoothness of the delivered streams, and how quickly clients are aggregated. The first of these variants is perhaps the most direct extension of the VBRBS technique as described in the previous section, and is referred to simply as VBRBS in the following. Whenever there is available client storage space for buffering more object data, the server delivers a stream at rate *crb*. Whenever the buffer space is filled, the server delivers a stream of rate given by the minimum of *crb* and the rate at which the buffer is being drained. This strategy minimizes the time required for a merger client to catch up with a mergee client, but may result in delivered streams with high bit rate variability.

The second of these variants, referred to as "VBRBS with smoothing" in the following, uses an "optimally" smoothed delivery schedule, computed by adapting the algorithm of Salehi et al.[21] as follows. The delivery schedule used by the first variant is the "upper bound" on the schedule to be computed. For the lower bound on the schedule, the "raw consumption" schedule is used, with two modifications. First, let L be the duration of the upper bound schedule. The duration of the raw consumption schedule is modified to equal L by supposing that all of the data consumed after L is instead consumed at that point. Second, this lower bound schedule is "minimally smoothed" in a single pass from the end of the schedule to the beginning, so that, at each point, its rate is less than or equal to *crb*.¶

Figure 10 compares the above two variants of VBRBS for the same four VBR objects considered in Section 5. The value of *crb* is fixed at 1.2, and the normalized client request rate is fixed at 200. The x-axis of each graph gives the available client storage capacity expressed as a fraction of the entire object. The "VBRBS" and "VBRBS (S)" (for "VBRBS with smoothing") curves give the required server bandwidth with these techniques for a start-up delay equal to that which would be required if the object was fully smoothed and delivered at the average object bit rate. The "VBRBS (R)" and "VBRBS (SR)" curves give the required server bandwidth for reduced start-up delay, specifically for the minimum possible start-up delay given the assumed constraint of *crb* on the transmission rate to the client. These results show that for small client storage capacity, there can be substantial server bandwidth cost to smoothing the delivered streams, as it may take longer for a delivered stream to "catch" a raw stream of an earlier client. For clients with limited storage, this must be traded off against the increased effectiveness of server and network resource allocation with smoothed streams.

¶If client storage space is insufficient to support this minimal smoothing, then either the client storage space or the achievable transmission rate to the client must be increased if streaming of the object is to be possible.



(a) Composite 1 (b) Composite 2

(c) Asterix (d) Dinosaur

Figure 10: Performance of VBRBS with Limited Client Storage Space ($N = 200$, $crb = 1.2$)

7. CONCLUSIONS

This paper has considered the interaction between two key technologies for efficient on-demand, real-time delivery of stored multimedia content: work-ahead smoothing, and recently proposed multicast delivery techniques that dynamically aggregate clients. It was shown that with aggressive smoothing (i.e., full smoothing to a constant bit rate stream), the straightforward approach of applying these technologies in tandem can be inefficient. This motivated development of a new delivery technique, called *VBR Bandwidth Skimming* (VBRBS), that adopts an integrated approach. Simulation results for a variety of VBR objects showed that for high client request rates and small "skims", the server bandwidth required by the previous techniques can be 50% or more higher than with VBRBS.

VBRBS achieves efficient client aggregation since it determines merge points based on the variable bit rate profile of the object being delivered, rather than on a smoothed stream that may include significantly more of the object data in its initial portion and thus take longer for a new client to catch up with. At the same time, VBRBS delivers fully-smoothed streams when sufficient client storage space is available. Effectively, the difference between the achievable transmission rate to the client, and the variable minimum rate at which the object must be delivered so as to avoid jitter, forms a variable "bandwidth skim" that can be used for catching up to earlier clients that have requested the same object.

On-going research includes further exploration of multicast delivery techniques for structured objects such as layered video and indexed multimedia presentations. Other current work concerns the experimental evaluation of various multimedia delivery and proxy caching techniques using a prototype system, and the integration of caching, error recovery, and security considerations into multimedia delivery systems.

REFERENCES

1. A. Bar-Noy, G. Goshi, R. E. Ladner, and K. Tam, "Comparison of stream merging algorithms for media-on-demand," in *Proc. IS&T/SPIE MMCN'02*, (San Jose, CA), Jan. 2002.
2. Y. Cai, K. A. Hua, and K. Vu, "Optimizing patching performance," in *Proc. IS&T/SPIE MMCN'99*, pp. 204–215, (San Jose, CA), Jan. 1999.
3. S. W. Carter and D. D. E. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. IEEE ICCCN'97*, pp. 200–207, (Las Vegas, NV), Sept. 1997.
4. E. G. Coffman, Jr., P. Jelenkovic, and P. Momcilovic, "Provably efficient stream merging," in *Proc.* 6^{th} *Int'l. Workshop on Web Caching and Content Distribution*, (Boston, MA), June 2001.
5. D. L. Eager, M. K. Vernon, and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," *IEEE Transactions on Knowledge and Data Engineering* **13**, pp. 742–757, Sept./Oct. 2001. (Earlier version appears in *Proc. MIS'99*.)
6. D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proc. ACM MULTIMEDIA'99*, pp. 199–202, (Orlando, FL), Nov. 1999.
7. D. L. Eager, M. K. Vernon, and J. Zahorjan, "Bandwidth skimming: A technique for cost-effective video-on-demand," in *Proc. IS&T/SPIE MMCN'00*, pp. 206–215, (San Jose, CA), Jan. 2000.
8. W. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," in *Proc. IEEE Infocom'97*, pp. 58–66, (Kobe, Japan), April 1997.
9. L. Gao and D. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," in *Proc. ICMCS'99*, **2**, pp. 117–121, (Florence, Italy), June 1999.
10. M. N. Garofalakis, Y. E. Ioannidis, and B. Ozden, "Resource scheduling for composite multimedia objects," in *Proc.* 24^{th} *VLDB*, pp. 74–85, (New York, NY), Aug. 1998.
11. M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SIGCOMM'94 Conf.*, pp. 269–280, (London, England), Aug. 1994.
12. A. Hu, "Video-on-demand broadcasting protocols: A comprehensive study," in *Proc. IEEE Infocom'01*, (Anchorage, AL), April 2001.
13. K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. ACM MULTIMEDIA'98*, pp. 191 – 200, (Bristol, U.K.), Sept. 1998.
14. K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. ACM SIGCOMM'97 Conf.*, pp. 89–100, (Cannes, France), Sept. 1997.
15. D. LeGall, "MPEG: A video compression standard for multimedia applications," *Communications of the ACM* **34**, pp. 46–58, April 1991.
16. A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel, "Scalable on-demand media streaming with packet loss recovery," in *Proc. ACM SIGCOMM'01 Conf.*, pp. 97–108, (San Diego, CA), Aug. 2001.
17. I. Nikolaidis, F. Li, and A. Hu, "An inherently loss-less and bandwidth-efficient periodic broadcast scheme for vbr video," in *Proc. ACM SIGMETRICS'00*, pp. 116–117, (Santa Clara, CA), June 2000.
18. J. F. Pâris, "A broadcasting protocol for compressed video," in *Proc. EUROMEDIA'99*, pp. 78–84, (Munich, Germany), April 1999.
19. J. F. Pâris, S. W. Carter, and D. D. E. Long, "A hybrid broadcasting protocol for video on demand," in *Proc. IS&T/SPIE MMCN'99*, pp. 317–326, (San Jose, CA), Jan. 1999.
20. O. Rose, "Statistical properties of mpeg video traffic and their impact on traffic modeling in atm systems," in *Proc. IEEE* 20^{th} *Conf. on Local Computer Networks*, pp. 397–406, (Minneapolis, MN), Oct. 1995.
21. J. D. Salehi, Z. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirement through optimal smoothing," in *Proc. ACM SIGMETRICS'96*, pp. 222–231, (Philadelphia, PA), May 1996.
22. D. Saparilla, K. Ross, and M. Reisslein, "Periodic broadcasting with VBR-encoded video," in *Proc. IEEE Infocom'99*, pp. 464–471, (New York, NY), March 1999.
23. J. Song, A. Dan, and D. Sitaram, "Efficient retrieval of composite multimedia objects in the JINSIL distributed system," in *Proc. ACM SIGMETRICS'97*, pp. 260 – 271, (Seattle, WA), June 1997.
24. S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *ACM Multimedia Systems Journal* **4**, pp. 197–208, Aug. 1996.