Dynamic Skyscraper Broadcasts for Video-on-Demand *

Derek L. Eager
University of Saskatchewan
Saskatoon, SK Canada S7N 5A9
eager@cs.usask.ca

Mary K. Vernon
University of Wisconsin – Madison
Madison, WI 53706
vernon@cs.wisc.edu

Abstract

Skyscraper Broadcasting is a recent statically scheduled broadcast technique for video-on-demand that addresses the network-I/O bottleneck to provide significantly superior performance over previous approaches. In this paper, we define a scheme for *dynamically* scheduling the objects that are broadcast on the skyscraper channels and show that yet another significant improvement in performance can be obtained. The dynamic broadcasting scheme is designed to provide all clients with the precise time at which their requested object will be broadcast, or an upper bound on that time if the delay is small. New segment size progressions are proposed that (1) facilitate dynamic scheduling, (2) simplify the server disk layout problem, and (3) support clients with inexpensive settops that can buffer only one or two frames. Finally, we develop a simple analytic formula that aids in selecting optimal skyscraper configurations for a given request arrival rate, set of object selection frequencies, and target system cost.

Preliminary simulation results show that 1) the dynamic scheme provides factors of two to ten improvement in mean client waiting time, or conversely delivers a given target average client waiting time with a significantly lower number of channels as compared with the statically scheduled system, 2) the dynamic system outperforms the static system with respect to variability in client waiting time, and 3) clients with inexpensive (diskless) settops can receive a reasonable level of service while clients with more expensive settops are provided with a high level of service that is relatively isolated from detrimental performance impact from the diskless clients, for the dynamic scheme that is defined in this paper.

*This research was partially supported by the National Science Foundation under Grant HRD-9896132, and by the Natural Sciences and Engineering Research Council of Canada under Grant OGP-0000264.

Technical Report #1375, C.S. Dept, UW-Madison, May 1998. Submitted for publication.

1 Introduction

The dominant cost in video-on-demand systems (e.g., Time Warner's test market system in Orlando [11] or Microsoft's prototype Tiger Video Fileserver [3]) is the server disk-I/O and network-I/O bandwidth required to deliver quick response to the end user. To address this problem, a series of recent papers have proposed Pyramid Broadcasting [13, 14], Permutation-Based Pyramid Broadcasting [1], and, most notably, Skyscraper Broadcasting [8]. The innovative idea in these schemes is that the data for each object is divided into fragments of increasing size, and these fragments are broadcast during predefined periods on separate channels. The periodic broadcast of the smallest fragment is most frequent, allowing new requests to begin playback quickly. Periodic broadcast of each larger fragment is scheduled on a different channel in a manner such that a client can always begin receiving the next larger fragment either during or immediately following the broadcast of a given fragment. Clients must be able to receive on two channels simultaneously and must be able to buffer a fragment that is received earlier than needed for playback. Since multiple broadcasts of a smaller fragment are scheduled for each broadcast of a larger fragment, clients that receive different small segment broadcasts batch together *during playback* (i.e., when they receive the same larger segment broadcast), in addition to batching while they wait for playback as in earlier schemes. Thus, fewer total channels are required to provide fast client response than in conventional schemes. Of the three schemes that propose and improve on this idea, the skyscraper broadcast technique offers the lowest latency while maintaining a reasonable required client buffer size.

The skyscraper broadcast scheme divides objects into two categories: *hot* and *cold*. Each hot object is assigned K channels on which its skyscraper segments are periodically broadcast, as described above. A second parameter, W, specifies the largest segment size, which in turn bounds the storage required in the client settops. The cold objects are broadcast on the remaining channels using a conventional channel allocation algorithm such as first-come first-serve. Batching of requests for cold objects occurs only while the requests are waiting to start playback.

This paper proposes several potential improvements in the proposed statically-scheduled skyscraper broadcasting scheme. In particular, we:

- define a scheme for *dynamically* scheduling the objects broadcast on the skyscraper channels, thus allowing a much larger number of objects to reap the cost/performance benefits of the skyscraper broadcasts; the dynamic scheme provides clients with the precise time at which their broadcast will begin, or an upper bound on that time if the delay is small,
- provide a cost/performance analysis of alternative segment size progressions for the skyscraper broadcasts,
- derive new segment size progressions for the (static or dynamic) skyscraper channels;

these progressions simplify the server disk layout problem and allow clients with inexpensive settops (i.e., very limited storage) to receive the skyscraper broadcasts,

- develop a simple analytic formula for first-cut estimates that can guide the selection of optimal dynamic skyscraper configurations for a given request arrival rate, set of object selection frequencies, and target system cost, and
- provide a preliminary evaluation of the performance benefit that can be obtained by dynamically scheduling the skyscraper channels.

One motivation for dynamic skyscraper scheduling is that recent evaluation of conventional broadcasting schemes has shown that periodic broadcast of the hot objects on isolated groups of channels is not necessarily superior to dynamically scheduling all objects on all available channels [12]. Another motivation is that there may not be a clear distinction between *hot* and *cold* objects in many video-on-demand systems; furthermore, the particular objects that happen to be most popular may change with the time of day or as new objects are installed. Dynamic scheduling of the skyscraper broadcasts may be beneficial for a potentially large set of *lukewarm* objects on a given server, providing a smooth increase in broadcast frequency as a function of current object popularity as well as being more responsive to dynamically changing object popularity. Finally, the unused small-segment broadcasts for a particular lukewarm or cold object might be reassigned to requests that are waiting to catch up with a different skyscraper broadcast, further reducing the response time for more popular objects.

The remainder of the paper is organized as follows. Section 2 reviews the statically scheduled skyscraper broadcast scheme and defines the new dynamically scheduled alternative evaluated in this paper. Section 3 explores the cost/performance trade-offs of new segment size progressions, and section 4 presents the analytic formula for first-cut selection of optimal channel configurations. The simulation results that compare the performance of the static and dynamic skyscraper schemes are presented in section 5, and section 6 provides the conclusions of this work.

2 Skyscraper Broadcasts

In section 2.1 we describe more precisely how the skyscraper channels are organized. In section 2.2 we identify particular sets of broadcasts that batch together in the original skyscraper system, and then we use these newly defined *transmission clusters* to develop a dynamically scheduled skyscraper broadcast scheme that provides clients with precise times at which their requested object will broadcast, or with an upper bound on that time in the case that the delay is small. The notation used in the remainder of this paper is defined in Table 1.

Parameter	Definition
n :	number of objects that are broadcast on skyscraper channels
C :	total number of channels devoted to skyscraper broadcasts
K :	number of channels per skyscraper broadcast
P :	progression of relative segment sizes on the skyscraper channels
W :	the largest segment size in a skyscraper broadcast
N :	number of groups of dynamic skyscraper channels
T :	total time to play an entire object
T_1 :	duration of a unit-segment broadcast
L :	total size of the object
r :	required object playback rate
S :	total number of unit-segments in an object playback

Table 1: Parameters of a Skyscraper Broadcast System.

2.1 Static Skyscraper Broadcasts

In the skyscraper broadcast scheme [8], K channels are assigned to *each* of the n most popular objects. Each of the K channels repeatedly broadcasts a distinct segment of the object at the required playback rate. The progression of *relative* segments sizes on the channels, namely $\{1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, 105, 105, ...\}$, is bounded by the parameter, W, and padded with W values up to length K, in order to limit the storage capacity required in the client settop.

Figure 1 illustrates the periodic broadcasts that occur on the channels assigned to a given object for $K = 8$ and $W = 12$. Note that repeated broadcast of the first unit-segment occurs on channel 0, repeated broadcast of the next two-unit segment occurs on channel 1, and so forth. The gray shading in the figure illustrates the sequence of broadcasts that a client receives if the client requests the object just prior to the gray unit-segment broadcast on channel 0. Note that the client receives the segment on channel 3 simultaneously with the sequence of segments on channels 1 and 2. Four units of data from channel 3 are buffered while the data from channel 1 and 2 are played, and then the buffered data plays while subsequent data is buffered and while the gap between reception on channels 4 and 5 is filled. The data received on channels 5 through 7 in this broadcast sequence can be played for the viewer as it is received.

Two other reception sequences are shown in the figure. One sequence is diagonally striped; the other is horizontally striped. Note that these two sequences share the same broadcasts on channels 3 and 4, while the first of these two sequences shares the broadcasts on channels 5 through 7 with the gray shaded sequence. A total of eleven units of data must be buffered by a client who receives the diagonally striped broadcast sequence; the

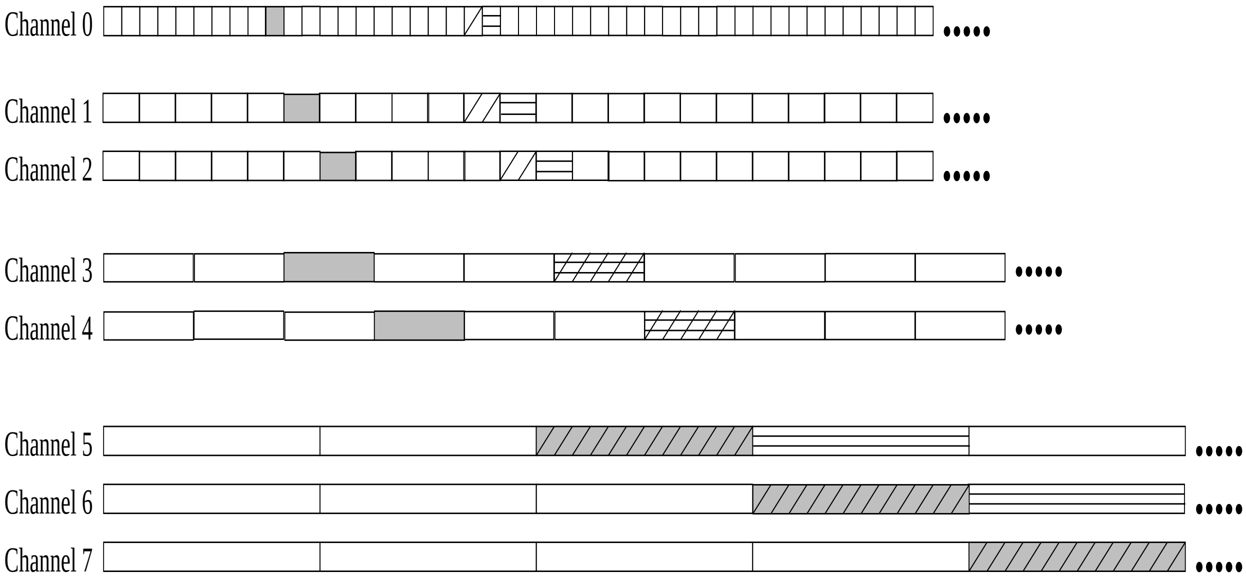

Figure 1: K Skyscraper Channels for Broadcasting a Single Object. ($K = 8$; $W = 12$)

horizontally striped sequence requires buffering of one unit-segment.

Hua and Sheu show that for the given sequence (and alignment) of relative segment sizes, a client starting in any unit-segment broadcast can receive the necessary sequence of segments without jitter, requiring simultaneous reception on at most two channels by the client. They also show that the required storage in the client settop is equal to $(W-1) \times L/S$, where L is the total size of the object, and S is the sum of the relative segment sizes that are repeatedly broadcast on each of the K channels. Note that L/S is the size of a unit-segment.

The duration of each unit-segment broadcast, T_1, is equal to the total object playback time (T) divided by S, where $T = L/r$ and r is the playback rate. As an example, if K=8, W=12, and the object has total playback duration equal to two hours, a new broadcast begins on channel 0 every 2.35 minutes. This can be contrasted with a conventional broadcast system, where each channel broadcasts an entire object. In this case, one would need to assign 24 channels to a two-hour object in order to have a playback begin every 5 minutes.

The reason this scheme was named skyscraper broadcasting is that when the segment sizes are stacked one above the other, they form the shape of a skyscraper.

The paper by Hua and Sheu does not discuss how VCR functions such as rewind and fast forward might be handled in a statically scheduled skyscraper scheme. We note that these functions might be supported fairly well by the segments that are broadcast on the K channels at any given point in time.

2.2 Dynamic Scheduling of Skyscraper Broadcasts

In this paper, rather than devoting K channels to repeated broadcast of the skyscraper segments for a single object, we investigate the performance potential of dynamically changing the object that is broadcast on the skyscraper channels, in response to client requests. We call a sequence of skyscraper segments that an individual client receives a *reception sequence*. A key question is how to identify reception sequences where it is safe to change the object that is broadcast. We initially assume that all objects have the same playback duration. Varying object lengths are discussed in section 2.2.4.

2.2.1 Skyscraper Transmission Clusters

We require non-overlapping clusters of skyscraper broadcast periods that can be dynamically scheduled. Let each non-overlapping transmission cluster *start with* the earliest reception sequence that contains a given broadcast period on channel K. Thus, the gray shaded sequence in Figure 1 starts a new transmission cluster. The horizontally striped sequence starts the next transmission cluster in that figure. The reader can verify that a third cluster begins on channel 0 twelve slots after the horizontally striped period. In fact, each new cluster begins on channel 0 precisely W slots after the previous sequence.

The other reception sequences that belong to a given transmission cluster (for example, the cluster that starts with the gray reception sequence in Figure 1) are all sequences that (1) use the same segment broadcast on channel K, and (2) do not use any broadcast periods on channels 0 through $K-1$ that are in (the first sequence of) the next transmission cluster. Thus, the cluster that begins with the gray reception sequence includes all of the clear segments between the gray shaded segment and the striped segments on channels 0 through 4, plus the diagonally striped segments on channels 1 and 2. The dynamic system will allow each cluster to broadcast a different object in response to client requests, rather than broadcasting the same object as the previous cluster. All sequences in the same cluster will broadcast the same object, allowing client requests to batch together during playback, as in the static skyscraper system.

The diagonally striped segment on channel 0 in Figure 1 starts a reception sequence that requires broadcast periods on channels 3 and 4 that are in the next transmission cluster. Thus, this segment on channel 0 is not a member of any transmission cluster, and it will not be used in a dynamically scheduled system that has the given segment size progression. We address this issue further in section 2.2.6.

The above-defined transmission clusters are key to designing an efficient dynamically scheduled skyscraper broadcast scheme, discussed next.

2.2.2 The Basic Dynamic Scheme

Let C channels in the system be allocated to the dynamically scheduled skyscraper broadcasts, and let these channels be organized into N groups of K channels each, where C and K

are parameters of the configuration. As in the static skyscraper system, broadcasts within each group of K channels have a fixed segment size progression across the channels in the group that is upper bounded by the parameter W. For now the reader should assume that the segment size progression is the same as defined for the static skyscraper system and that broadcast periods within a group of channels are aligned as in Figure 1. Thus, each group of K channels broadcasts a sequence of transmission clusters that begin every W slots on channel 0.

The transmission clusters in the different groups of channels are *persistently staggered* such that a new transmission cluster starts on a different group every $\tau = \frac{W \times T_1}{N}$. Each transmission cluster can be scheduled to broadcast any object, in response to client requests. A server disk layout that makes this possible is briefly discussed in section 3.

A new client request is scheduled as follows. First, the client is assigned to the next unit-segment broadcast of a transmission cluster that has already been assigned to that object, if any. Otherwise, the client is assigned to the next unit-segment broadcast of a transmission cluster that has already started but hasn't yet had an object assigned, if any. Finally, the request is scheduled on the next available transmission cluster that will begin broadcasting in the future.

Requests that require a new transmission cluster are scheduled in first-come first-serve (FCFS) order for two reasons. First, recent results show that for fixed length objects, FCFS outperforms other scheduling algorithms such as *maximum queue length first* (MQL) or *maximum factored queue length first* (MFQ) if both the mean and the variability in client waiting time are considered [12]. Second, the broadcast assignment can be done when the request arrives, and thus the system can immediately inform the client when the broadcast will begin.

2.2.3 Temporary Channel Stealing

Several optimizations of the above dynamic skyscraper scheme are possible. For example, a unit-segment broadcast period on channel 0 that is not needed to serve new requests for the object can be reassigned to requests that are waiting for a unit-segment broadcast in another active transmission cluster. This is possible because each transmission cluster can serve any object. The clients that can be served by temporary channel 0 reassignments were given a short broadcast delay (i.e., less than T_1), which should be reported as an upper bound rather than a precise delay if channel 0 reassignment is implemented. Note that the requests in an active transmission cluster can only be served early if (1) the two-unit broadcast on channel 1 in their group will begin at the same time as the next unit-segment broadcast in their group, or (2) if channel 1 in the transmission cluster that is doing the stealing is ready to broadcast a two-unit segment and can also be temporarily reassigned. In the case that a channel 1 period is reassigned, the subsequent channel 2 broadcast can also be reassigned, which simplifies the reception sequence for the clients that are served early.

We implement the reassignment of channels 0 and 1/2 described above in the simulator that is used to evaluate the dynamic skyscraper scheme in section 5. When a channel 0 broadcast is reassigned, it is assigned to the eligible request that has been waiting the longest over all other active transmission clusters. Furthermore, when a new client request is assigned to an idle transmission cluster that is already in progress, it is assigned to the cluster that has the longest remaining time until its next unit-segment broadcast, so that the new request is not served ahead of requests that are waiting for other unit-segment broadcasts. This policy turns out to provide noticeably improved performance when temporary channel reassignment is implemented.

It is also possible to reassign unused broadcast periods on higher numbered channels to requests waiting in another transmission cluster, but this further improvement in stealing is beyond the scope of this paper.

2.2.4 Varying Length Objects

A particular video-on-demand system may have objects of different lengths, such as two-hour movies, thirty-minute television shows, fifteen minute news summaries, and three-minute news clips. In a campus environment, the server might contain two-hour movies, one-hour lectures, and fifteen-minute lab presentations. In such environments, we envision that objects will be partitioned by object size and playback rate, and each class of objects will be assigned a fraction of the total available channels (based on the relative demands for each class). Some or all of the channels for a given class would then be configured for dynamically scheduled skyscraper broadcasts. This approach can be used to achieve particular performance objectives, such as waiting time proportional to object playback duration. Allocating channels among different object sizes is an important topic for future research.

2.2.5 VCR Functions

VCR functions in the dynamic skyscraper system could be partially supported by the segments broadcast in a transmission cluster, and otherwise by background processing as proposed in the Tiger system [3], with extra contingency channels as proposed by Dan et. al. [4], and/or by buffering segments in the client settop explicitly for this purpose.

2.2.6 The Segment Size Progression Problem

The segment size progression proposed by Hua and Sheu for statically scheduled skyscraper broadcasts, $\{1, 2, 2, 5, 5, 12, 12, ...\}$, appears to have the maximum possible increases in relative segment size on the K channels, subject to two constraints: (1) for any initial unit-segment broadcast, there must be a sequence of segments that the client can receive that will support continuous playback to the viewer, and (2) clients are required to receive data on no more than two channels simultaneously. Maximum increases in segment size are desirable because this results in smaller unit segments, which in turn increases the frequency at which broadcasts begin on channel 0. On the other hand, the proposed progression of segment sizes is not ideal for the dynamically scheduled skyscraper system because particular segment broadcasts can't be used in any transmission cluster. For example, one would like to include the diagonally striped segment on channel 0 in Figure 1 in the transmission cluster that starts with the gray shaded sequence, but that reception sequence requires a broadcast on channel 3 that is needed in the next transmission cluster. This does not pose a problem in the static skyscraper system because each channel in a group of K channels is broadcasting the same segment in every transmission cluster. However, in the dynamic system different transmission clusters on a given group of channels may be broadcasting different objects.

To address the problem of unused channel bandwidth, we investigate the cost/performance implications of alternative segment size progressions in the next section.

3 Alternative Segment Size Progressions

Section 3.1 explores new segment size progressions that avoid broadcast conflicts between transmission clusters, thus allowing all reception sequences to be used in the dynamic system. Sections 3.2 and 3.3, respectively, comment on two additional advantages of the new progressions for static or dynamic skyscraper systems: (1) they provide service for clients that have inexpensive settops with storage capacity for only one or two frames, and (2) they simplify the server disk layout problem.

3.1 Conflict-Free Transmission Clusters

We consider relative segment size progressions of the form $\{1, a, a, b, b, c, c, ...\}$, upper bounded by the parameter W. This is the basic structure of the progression proposed by Hua and Sheu. A key observation is that the width of a transmission cluster on channels 0 and K is equal to W. Thus, to avoid conflicts or holes between transmission clusters the width of the transmission cluster on channels 1 through $K-1$ must also be equal to W. (Note that in Figure 1 the width of a transmission cluster on channels 3 and 4 is not equal to W, which causes the conflict between the striped segments on these channels.) A necessary condition for transmission group widths equal to W is that *each relative segment size must evenly divide all higher relative segment sizes*. Candidate sequences include: $A = \{1, 2, 2, 4, 4, 8, 8, 16, 16, ...\}$, $B = \{1, 2, 2, 6, 6, 12, 12, 24, 24, ...\}$, and $C = \{1, 2, 2, 6, 6, 12, 12, 36, 36, ...\}$. Progressions A and B/C are illustrated in Figure 2.

We claim that the following additional requirements guarantee conflict-free transmission clusters as well as jitter-free reception starting from any unit-segment broadcast on channel 0:

- the relative segment size on channels 1 and 2 is two (i.e., $a = 2$),
- the segment size increases by *at most a factor of three* at each other step in the progression, and

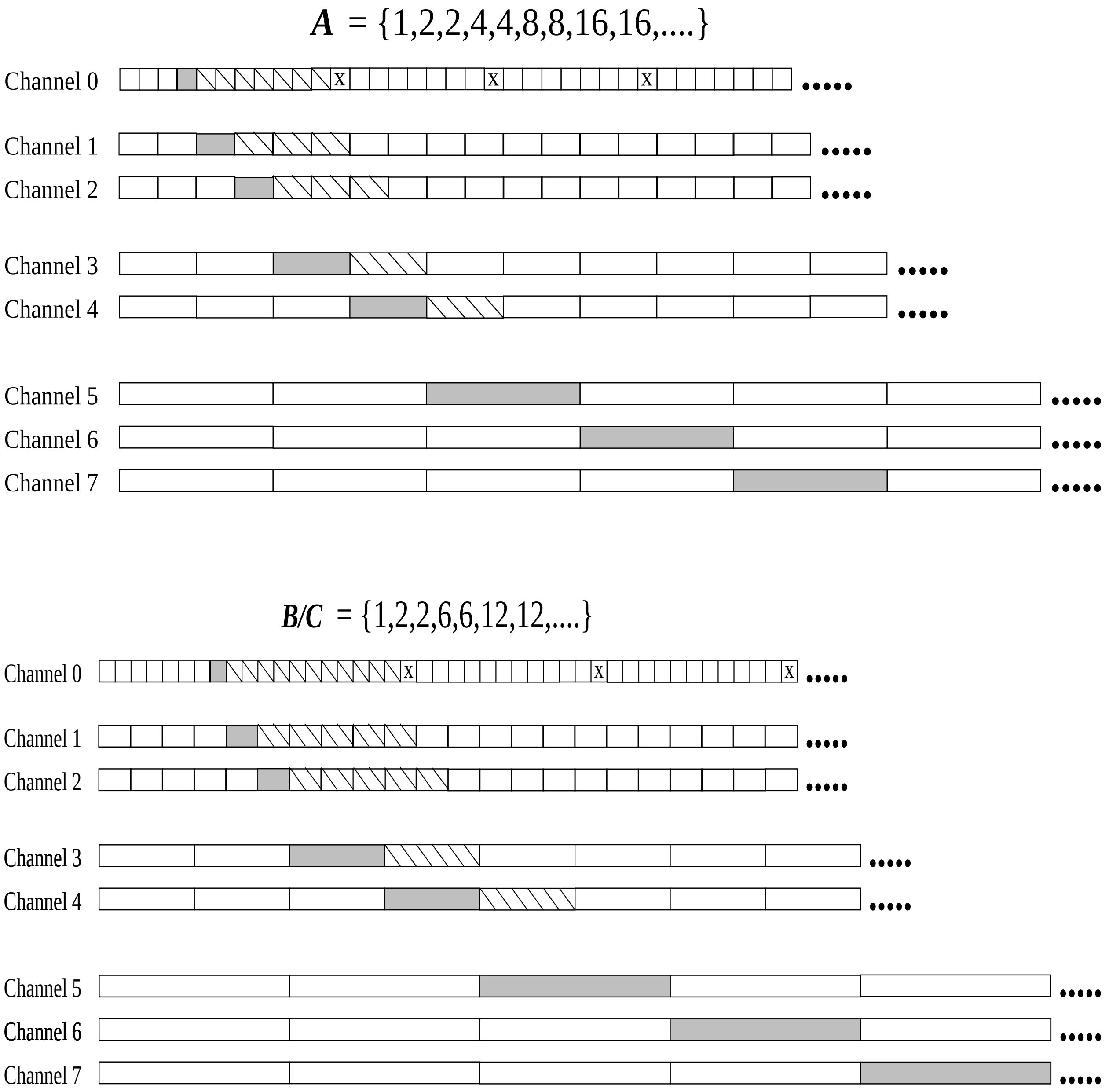

Figure 2: Alternative Segment Size Progressions. ($K = 8$; $W = 8$ for A, $W = 12$ for B/C)

- the transmission cluster of width W on a given channel $k > 0$ starts just after channel $k-1$ broadcasts its first segment of the transmission cluster.

Referring to Figure 2, the argument for the above claim is as follows. Due to the third condition, the reception sequence that starts in the first broadcast period on channel 0 for any transmission cluster is jitter-free and requires reception on only one channel at a time. For each other reception sequence in the transmission cluster:

- due to conditions one and three, the reception on channel 1 either directly follows the reception on channel 0 or is overlapped with it,

- for an odd-numbered channel i, the reception on channel $i+1$ immediately follows the reception on channel i since these two channels have the same segment size,

- for an odd-numbered channel i, if $i+2$ broadcasts segments twice as large (e.g., progression A), then the broadcast on channel $i+2$ is either aligned with the end of the reception on channel i or the end of the reception on channel $i+1$; if $i+2$ broadcasts segments three times as large (e.g., channel 3 in progression B or C), the broadcast on $i+2$ is aligned with the start of the broadcast on channel i, the end of the reception on channel i, or the end of reception on channel $i+1$.

We further claim that progression A is the fastest increasing progression that avoids holes and conflicts between transmission clusters and that also requires simultaneous reception on at most two channels. Any increase by a factor of 3 in the progression requires (1) the client to receive on three channels simultaneously for some of the reception sequences, and (2) temporary reassignment of more than channels 0, 1, and 2 for early service in some cases.[1] We claim without further proof that progressions B and C require reception on at most three channels simultaneously.

Note that the storage requirement for the new segment size progressions can be derived from Figure 2 by observing that the last unit-segment broadcast in a transmission cluster occurs $W-1$ time units after the first unit-segment broadcast. Thus, the clients that begin in the last unit-segment broadcast will receive segments $W-1$ units ahead of their playback time once they batch with the clients that start in the first unit-segment broadcast. Also note that while larger W implies a larger storage requirement, it also implies smaller T_1, which implies smaller average and maximum client waiting time. Tables 6 and 7 in section 4 illustrate the storage requirements and unit-segment playback durations for two-hour MPEG-1 movies (having playback rate of 1.5×10^6 bits per second) for progression A under various channel configurations (K and W).

In summary, the cost analysis shows that progression A and the progression defined by Hua and Sheu each require reception on at most two channels, whereas progressions B

[1] In the simulations in section 5, only the requests that can be served early by channel 0 or by channels 0, 1, and 2, are eligible for channel stealing.

and C require reception on at up to three channels. Progression A has somewhat smaller segment size increases than the progression defined by Hua and Sheu and thus might have slightly higher expected waiting time if used in a static skyscraper system. However, we show next that the new progression(s) simplify server disk layout and accommodate clients with inexpensive settops. Furthermore, progression A will have better performance than the Hua and Sheu progression in a dynamic skyscraper system because no channel bandwidth is wasted. Progressions B and C have larger segment size increases, and thus would have better performance than the Hua and Sheu progression in either the static or dynamic system, at the cost of an extra tuner (and some extra storage) in each client settop.

3.2 Heterogeneous Clients

One advantage of the new segment size progressions considered above is that the first reception sequence in each transmission cluster requires very little buffering in the client settop. That is, the client only needs to receive on one channel at a time and must only buffer the frame that it is currently receiving. Thus, a skyscraper system that uses one of the new progressions can provide service to clients with inexpensive (diskless) settops as well as significantly better service to clients that have the storage capacity for catching up in a transmission cluster (typically provided by a standard commodity disk). We provide results in section 5 to illustrate the two levels of service. Note that it is difficult to find reception sequences in the original skyscraper segment size progression (Figure 1) that allow the client to receive on just one channel at a time and thus buffer only one or two frames.

3.3 Simplified Server Disk Layout

One key design issue for multimedia storage servers is how to efficiently store and retrieve data on the server disks in order to minimize system cost for a given number of channels. A possibly important advantage of the new segment size progressions defined in section 3.1 is that the data layout on the server disks is simplified, as explained below.

Previous papers have shown that coarse-grained striping of the data for each object across the disks achieves high performance by balancing the load across the disks [2, 7, 5, 10, 3]. For the case where all objects have the same fixed playback rate, Ozden et. al. [10] show how to compute the optimal number of disks and the optimal stripe unit size to minimize cost for a desired number of concurrent channels.

For the skyscraper broadcasts with the new segment sizes, each sequence of segments of total length W that will be broadcast in a transmission cluster on a given channel $k \leq K$ forms a set of data that can be striped across the disks. We define a simple modification to the minimum cost disk configuration that ensures that the needed data will be available for the next transmission cluster. That is, the stripe unit size is adjusted, or the length of the object is slightly padded (e.g., with advertising) so that the number of stripe units is an integer multiple of the number of disks. If the disks are numbered 1 through D and the first stripe unit of each object is placed on disk 1, the last stripe unit is always on disk D. Thus, whenever a channel is broadcasting the last stripe unit in a given transmission cluster, the first unit of the object to be broadcast in the next transmission cluster is always on the next disk that will supply data for the channel. Note that if the number of stripe units equals $s \times D$ and if the number of segments broadcast per transmission group on a given channel is j, then if j evenly divides s, a single segment of the data for that channel will be evenly striped across the disks; in this case no replication of the segment on the disks is required. More generally, $\frac{j}{GCD(j,s)}$ replications of the segment will be evenly striped across the disks since each segment is of length $\frac{sD}{j}$ stripe units. Thus, only $\frac{j}{GCD(j,s)}$ replications of the segment should be striped across the disks, rather than W replications.

4 Optimal Channel Configurations

Hua and Sheu do not provide criteria for selecting optimal channel configurations for the static skyscraper system. In this section, we provide a simple analytic formula that can be used as a *first cut* in identifying promising channel configurations for dynamic skyscraper systems. Segment size progression A defined in section 3.1 is used to illustrate the analysis.

Let C^* be the number of channels at the knee of the curve of mean client waiting time versus the inverse of the number of channels (see Figure 3). Letting $C^* = K \times N^*$ and λ_i denote the rate of requests for object i, we use a result from asymptotic bounds analysis of queueing networks [9] to estimate N^*:

$$N^* \approx \sum_{i=1}^{n} \frac{WT_1}{(W-1)T_1 + \frac{1}{\lambda_i}}$$

(Recall from Table 1 that n is the number of objects that can play on the skyscraper channels.)

The intuition behind the above formula is as follows. For each term (or object) in the sum, the numerator is the duration of a transmission cluster on each channel that is used to transmit the object, while the denominator is the average time between requests for a new transmission cluster for the object. (Note that $1/\lambda_i$ is the average time between requests for object i.) Thus, the ratio is the average number of groups of channels that would be in use by object i if an infinite number of groups were available. Summing over all objects that use skyscraper broadcasts gives an estimate of the total number of groups that should be provided.

Figure 3 illustrates how close the above estimate of C^* is to the knee of the curve for two pairs of values of K and W, given that 224 objects can be scheduled on the skyscraper channels, the total request arrival rate for those objects is eight requests per minute, and the object selection frequencies are given by the Zipf(0) distribution on 1000 objects.[2] Note

[2]We used n=224 because if there are 1000 objects in the system and the object selection frequencies are

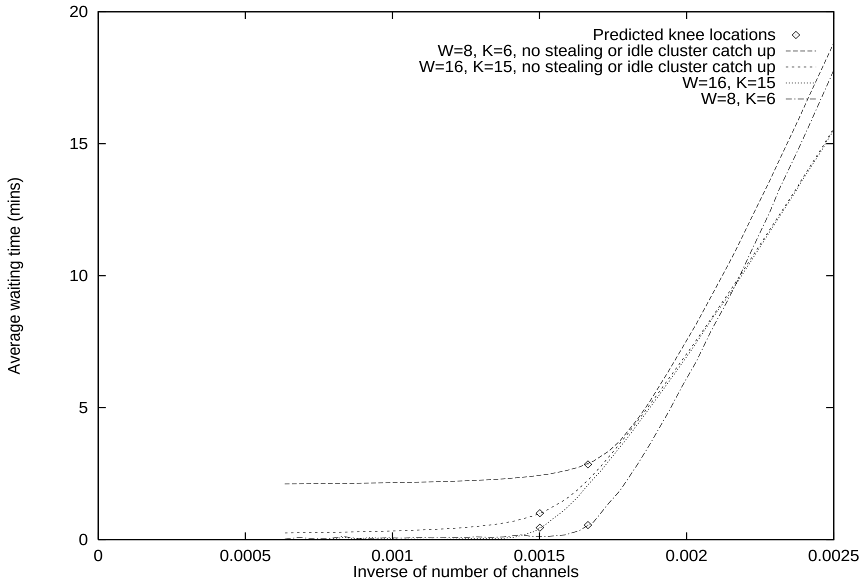

Figure 3: Average Client Waiting Time vs. Inverse of Number of Channels.
(224 120-minute objects; $\lambda = 8.0$)

that the C^* estimates are fairly close to the knees of the curves. In particular, the estimates appear to be *conservative*; that is, it appears to be important to avoid having fewer than C^* channels in each of the given configurations. The figure also illustrates the substantial performance improvement that is obtained near the knees of the curves due to (1) channel stealing and (2) allowing new client requests to *catch up* with idle transmission clusters.

Tables 2 and 3 give the computed C^* estimates as a function of K and W for a system with 1000 objects, client request arrival rate equal to ten or forty requests per minute, selection frequencies modeled by the Zipf(0) distribution, and object playback duration equal to 120 minutes. Tables 4 and 5 give the computed C^* estimates for the case where only the 224 most popular objects in the system (i.e., 80% of the client requests) use the skyscraper channels.

Although estimates of object selection frequencies that will occur in future video-on-demand systems are essentially impossible to obtain, the tables illustrate how the C^* estimates can guide system sizing and the selection of optimal channel configurations for various

modeled by the Zipf(0) distribution, the most popular object is selected with probability 13% and the 100th most popular object is selected with probability 0.0013; this distribution matches reasonably well with a particular measurement of the selection frequencies of the 100 most popular objects in a video rental outlet [6]. In this case 80% of the requests are for the 224 most popular objects, which is a popular definition of the hot set.

W / K	2	4	8	16	32	64	128
3	1524						
4	1676	1936					
5	1795	1985					
6	1896	2034	2532				
7	1981	2086	2478				
8	2056	2136	2456	3168			
9	2124	2178	2457	3015			
10	2180	2220	2460	2930	3830		
11	2233	2266	2475	2871	3586		
12	2280	2304	2484	2832	3432	4524	
13	2327	2340	2509	2808	3315	4186	
14	2380	2380	2520	2800	3248	3948	5236
15	2415	2400	2535	2790	3180	3795	4800
16	2448	2432	2560	2784	3152	3664	4496
17	2482	2465	2584	2788	3111	3570	4267
18	2520	2502	2592	2790	3096	3510	4104
19	2546	2527	2622	2793	3078	3458	3971
20	2580	2540	2640	2800	3060	3400	3880

Table 2: C^* Estimates.
(1000 120-minute Objects; $\lambda = 40.0$)

W / K	2	4	8	16	32	64	128
3	630						
4	660	840					
5	685	815					
6	708	804	1092				
7	728	805	1015				
8	744	808	976	1360			
9	765	810	954	1242			
10	770	820	940	1160	1640		
11	792	825	924	1111	1474		
12	804	828	924	1080	1368	1944	
13	806	832	923	1053	1287	1716	
14	826	840	910	1036	1246	1582	2254
15	825	855	915	1020	1200	1470	1965
16	848	848	912	1024	1168	1408	1792
17	850	867	918	1003	1139	1343	1666
18	864	864	918	1008	1134	1314	1566
19	874	874	912	1007	1121	1273	1501
20	880	880	920	1000	1100	1240	1440

Table 3: C^* Estimates.
(1000 120-minute objects; $\lambda = 10.0$)

hypothetical, projected, or measured object selection frequencies and client arrival rates. Since the C^* estimates are conservative, candidate optimal configurations for a given total number of channels for skyscraper broadcasts, C, are those values of K and W for which $C^* \leq C$. Note that, for smaller numbers of channels, the optimal configuration has smaller W which is closer to conventional (FCFS) broadcasts. Note also that a very large number of channels is required to use skyscraper broadcasting for 1000 objects with arrival rate equal to 40 requests per minute and the Zipf(0) distribution. This reflects the large number of objects of long playback duration, as well as the heavy tail of the Zipf distribution (e.g., in the Zipf(0) distribution, 20% of the requests are for the *least popular 776 objects*). C^* is significantly lower for less aggressive systems that have fewer or shorter objects, or with object selection frequences that have less mass for the least popular objects.

The first-cut estimates of C^* can be compared against client storage requirements (illustrated in Table 6 for two-hour MPEG-1 objects) and against T_1, the unit segment transmission time (illustrated in Table 7 for two-hour objects). Recall that the duration of a unit-segment broadcast is an upper bound on the waiting time for a request that can be scheduled in an active transmission cluster. Note that an increase in K decreases both the unit-segment broadcast duration and the required storage capacity, but also reduces the total number of groups of channels for a given number of available channels. An increase in W improves unit-segment waiting time but also increases required storage capacity.

W	2	4	8	16	32	64	128
K							
3	630						
4	768	776					
5	875	875					
6	972	954	1038				
7	1050	1029	1099				
8	1120	1088	1152	1304			
9	1188	1143	1197	1341			
10	1240	1200	1240	1370	1580		
11	1298	1243	1287	1397	1584		
12	1344	1284	1320	1428	1596	1872	
13	1391	1326	1365	1456	1612	1846	
14	1428	1372	1400	1484	1624	1834	2170
15	1470	1410	1425	1500	1635	1830	2115
16	1504	1440	1456	1536	1664	1840	2096
17	1547	1462	1479	1564	1666	1836	2074
18	1584	1494	1512	1584	1692	1836	2052
19	1615	1520	1539	1596	1710	1843	2052
20	1640	1560	1560	1620	1720	1860	2040

Table 4: C^* Estimates.
(224 120-minute objects; $\lambda = 32.0$)

W	2	4	8	16	32	64	128
K							
3	399						
4	432	468					
5	460	480					
6	486	492	606				
7	504	504	595				
8	528	520	592	752			
9	540	531	594	720			
10	550	540	590	700	910		
11	572	550	594	693	858		
12	576	564	600	684	816	1068	
13	585	572	611	676	793	1001	
14	602	574	616	672	784	938	1246
15	615	585	615	675	765	915	1140
16	624	592	624	672	752	880	1072
17	629	595	629	680	748	867	1020
18	630	612	630	684	738	846	990
19	646	608	627	684	741	836	950
20	660	620	640	680	740	820	940

Table 5: C^* Estimates.
(224 120-minute objects; $\lambda = 8.0$)

5 Experimental Results

This section provides some preliminary results from simulation experiments to illustrate the potential of the dynamically scheduled skyscraper system. A complete comparison of the dynamic and static skyscraper systems would be based on optimal configurations of each for different values of request arrival rates, total number of objects on the server, sizes of the objects, and object selection frequencies. Parameters of the optimal configuration for each point in this very large design space include the number of hot objects (static system), the partitioning of objects into an optimal number of classes (dynamic system), the channel partitioning between hot/cold or classes of objects, and the optimal values of K and W for each object or class of objects that uses skyscraper broadcasts. The optimization model that would support such a complete comparison is beyond the scope of this paper.

Instead, we provide some initial comparisons of the static and dynamic systems for a few points in the design space, using experimentally determined optimal configurations that satisfy particular constraints. As in section 4, we provide results for the aggressive system with 1000 objects each with a 120 minute playback duration, and Zipf(0) selection frequencies. The results are qualitatively similar for less aggressive systems. The simulation results for average client waiting time have 95% confidence intervals that are within 10% of the reported values for Figures 4 and 6, and are otherwise within 2% of the reported values.[3]

[3]Figures 4 and 6 were generated using an optimization procedure in which large numbers of candidate configurations were simulated.

W / K	2	4	8	16	32	64	128
3	270						
4	193	450					
5	150	312					
6	123	239	450				
7	104	193	326				
8	90	162	256	450			
9	80	140	210	332			
10	72	123	179	263	450		
11	65	110	155	218	335		
12	59	99	137	186	267	450	
13	54	90	123	162	222	337	
14	50	83	112	144	190	269	450
15	47	77	102	129	166	224	337
16	44	72	94	118	147	192	270
17	41	67	87	108	133	168	225
18	39	63	81	99	120	149	192
19	37	59	76	92	110	134	168
20	35	56	72	86	102	122	150

Table 6: Client Storage Requirement (MB). (object length, L = 1350 MB)

W / K	2	4	8	16	32	64	128
3	24.0						
4	17.14	13.33					
5	13.33	9.23					
6	10.91	7.06	5.71				
7	9.23	5.71	4.14				
8	8.0	4.8	3.24	2.67			
9	7.06	4.14	2.67	1.97			
10	6.32	3.64	2.26	1.56	1.29		
11	5.71	3.24	1.97	1.29	0.96		
12	5.22	2.93	1.74	1.1	0.76	0.63	
13	4.8	2.67	1.56	0.96	0.63	0.47	
14	4.44	2.45	1.41	0.85	0.54	0.38	0.31
15	4.14	2.26	1.29	0.76	0.47	0.31	0.24
16	3.87	2.11	1.19	0.69	0.42	0.27	0.19
17	3.64	1.97	1.1	0.63	0.38	0.24	0.16
18	3.43	1.85	1.03	0.59	0.34	0.21	0.13
19	3.24	1.74	0.96	0.54	0.31	0.19	0.12
20	3.08	1.64	0.9	0.51	0.29	0.17	0.1

Table 7: Unit-Segment Broadcast Duration (T_1). (120-minute objects)

5.1 Principal Performance Comparison

We consider systems in which objects are divided two classes: the k most popular *hot* objects and the other $1000 - k$ *cold* objects. Each possible division ($0 \leq k \leq 1000$) is considered, so that our results cover cases of hot sets containing only a few of the very hot objects, as well as hot sets including many *lukewarm* objects. The available channels are statically partitioned between the two classes. We then consider three combinations of broadcast techniques: (1) static skyscraper broadcasts for each object in the hot set and conventional FCFS (with *persistent staggering* of the allocated channels) for the cold objects, (2) dynamic skyscraper broadcasts for the set of hot objects and conventional FCFS for the cold objects, and (3) dynamic skyscraper broadcasts for each set of objects.[4] The relative segment size progression $\{1, 2, 2, 4, 4, 8, 8, 16, 16, ...\}$ is used for the dynamic skyscraper broadcasts, whereas the slightly higher-performance original skyscraper progression, $\{1, 2, 2, 5, 5, 12, 12, 25, 25, ...\}$, is used for the static skyscraper broadcasts.

Performance results are presented for experimentally determined optimal configurations. That is, for each considered division into hot and cold object sets, a search is performed for a channel partitioning that minimizes the overall average client waiting time. For the static skyscraper scheme, the number of channels allocated to each hot object, K, is equal to the number of channels allocated to the hot set divided by the size of the hot set. The parameter W is selected to be the largest possible given the derived K, so as to provide the most favorable performance data for the static skyscraper system. For the dynamic

[4] Recall that, among the channel scheduling policies that have been proposed to date for conventional broadcasts, FCFS has the best performance when both mean and variability in client waiting time are considered [12].

skyscraper scheme, the only constraint on K is that it evenly divide the number of channels assigned to the hot or cold set. A search is performed for the values of K and W that yield the lowest value for the sum of average client waiting time plus maximum observed client waiting time for the given class of objects.[5]

Figure 4 gives the results for average client waiting time, for a system with 1000 objects each with a 120 minute playback duration and Zipf(0) selection frequencies, 1600 channels in total, and an arrival rate of 40 requests per minute. Comparing the curve for the static skyscraper/FCFS combination with the curve for the dynamic skyscraper/FCFS combination, it is apparent that for small hot set sizes, the average waiting time is very similar in each system. In this and other experiments, we have found that static skyscraper and dynamic skyscraper yield very similar performance when applied to small hot sets with high request arrival rates. As the hot set size increases, however, the performance of the static skyscraper/FCFS combination begins to deteriorate. In contrast, the dynamic skyscraper scheme is able to effectively schedule requests for large hot sets including many lukewarm objects. Further, overall system performance improves substantially, beyond that possible with static skyscraper/FCFS, as the hot set size is increased. The best performance with the dynamic skyscraper/FCFS combination is achieved in the limiting case of a hot set size of 1000, when all channels are used for dynamic skyscraper scheduling for all objects.

The use of dynamic skyscraper scheduling on both the hot and cold sets (in general, with different optimal values of K and W for each set), is seen to yield the best mean client waiting time for all of the considered divisions between hot and cold sets. With hot set sizes of five to twenty objects, the average waiting time is less than 30% of the lowest average waiting time with any hot set size for the static skyscraper/FCFS system. At the optimal operating point, the average client waiting time is also about 60% of the lowest average waiting time in the dynamic skyscraper/FCFS combination.

Figure 5 shows that the use of dynamic skyscraper scheduling for both the hot and cold sets also yields improved performance with respect to the maximum waiting time observed in the simulations. With a hot set size of 5, the maximum waiting time observed when dynamic skyscraper is used on both the hot and cold sets is less than 40% of the lowest maximum waiting time observed for any hot set size for the static skyscraper/FCFS combination. Note that, in this case at least, the static skyscraper/FCFS combination does not improve maximum waiting time over that experienced with pure FCFS scheduling (the latter is shown in the figure by the results for a hot set size of 0).

Figure 6 gives the mean wait for cold objects and the mean wait for hot objects in the static skyscraper/FCFS system and in the system where dynamic skyscraper scheduling is used for both the hot and cold sets. Note that the average wait experienced by requests for hot and cold objects are relatively similar when dynamic skyscraper is used for both,

[5]Note that using the sum of mean and maximum waiting time for the objective function when searching for the optimal partitioning of channels between the hot and cold classes produced configurations that had significantly lower mean waiting time for cold objects as compared with hot objects, so we simply used mean waiting time as the objective function for determining that configuration parameter.

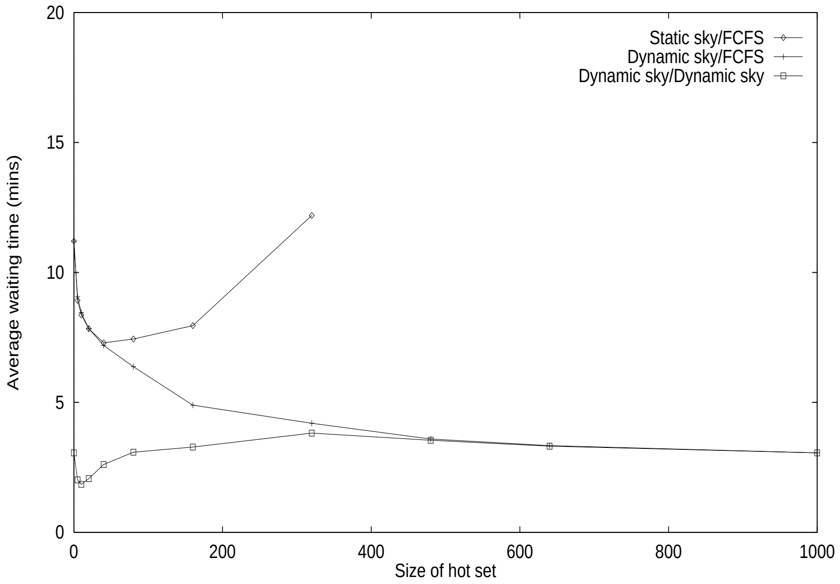

Figure 4: Average Client Waiting Time vs. Hot Set Size.
(1000 120-minute objects; 1600 channels; $\lambda = 40$)

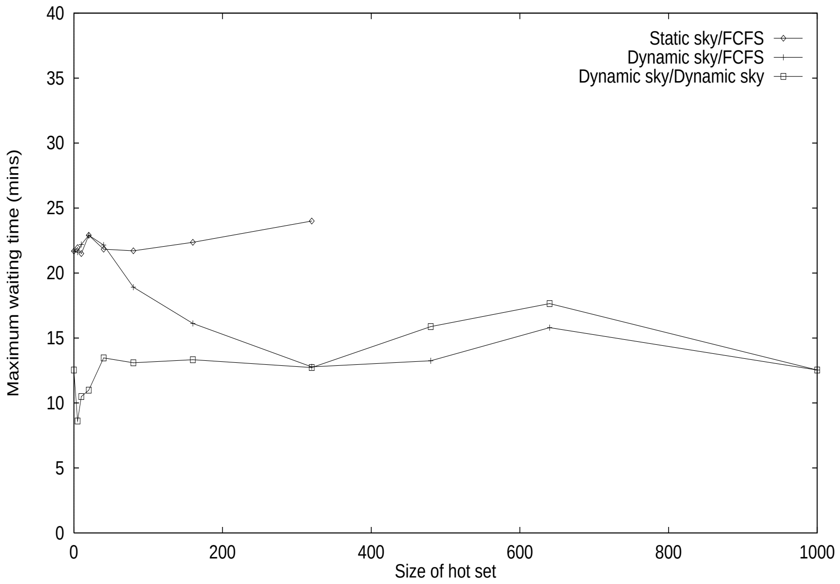

Figure 5: Maximum Client Waiting Time vs. Hot Set Size.
(1000 120-minute objects; 1600 channels; $\lambda = 40$)

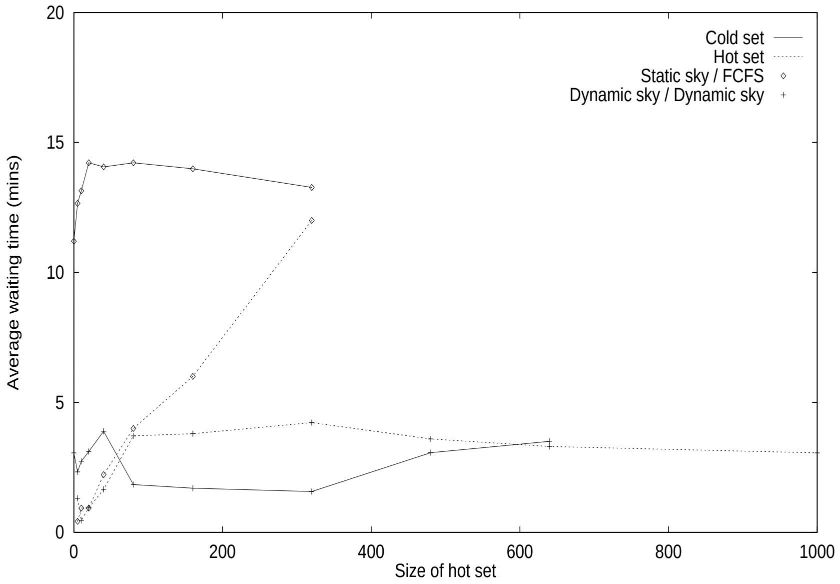

Figure 6: Average Waiting Time For Hot and Cold Objects vs. Hot Set Size.
(1000 120-minute objects; 1600 channels; $\lambda = 40$)

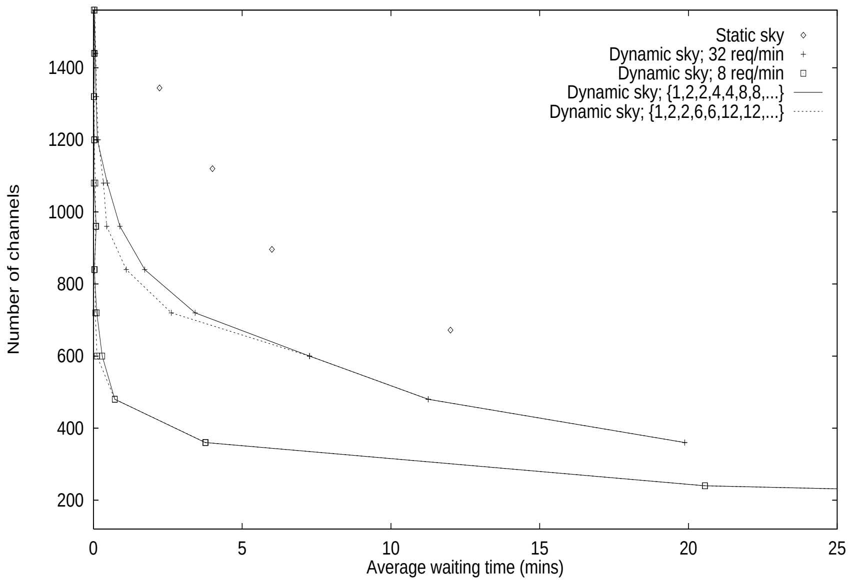

Figure 7: System Cost vs. Average Client Waiting Time.
(224 120-minute objects; $\lambda = 8$ or 32)

in contrast to the behavior with the static skyscraper/FCFS combination. Thus, use of dynamic skyscraper scheduling on both the hot and cold sets can also offer substantially improved fairness in comparison to the static skyscraper/FCFS combination.

5.2 Cost/Performance Comparison of Static vs. Dynamic Skyscraper Broadcasts

The performance advantages of the dynamic skyscraper scheme, shown above, result from the greater ability of the dynamic scheme to effectively schedule sets of objects that include objects of relatively low popularity. This is illustrated in Figure 7, which makes a direct comparison of the cost/performance tradeoffs of static vs. dynamic skyscraper as applied to the set of the 224 most popular objects, assuming 1000 objects in total and a Zipf(0) distribution of object selection frequencies.

Average waiting time is shown on the X-axis while the number of channels required to attain the average client waiting time for this object set is shown on the Y-axis. The results show that for a large object set such as that considered in Figure 7, dynamic skyscraper has substantially lower cost for a given target performance. Conversely, for a given number of available channels, the dynamic scheme provides factors of two to ten improvement in mean waiting time.

Results are shown for dynamic skyscraper for both the segment size progression of {1,2,2,6,6,12,12,36,36,...} and for the base case progression of {1,2,2,4,4,8,8,16,16,...}. The more aggressive segment size progression provides some improvement, principally for high load and low target mean waiting time, but the improvement is relatively small for the cases considered.

5.3 Variability in Client Waiting Time

The above results have considered average and maximum waiting time, over all objects, or all objects within each of the hot and cold sets. Measures of the distribution of waiting time, on a per-object basis, are shown in Figure 8 for the system with dynamic skyscraper broadcasts for both hot and cold object sets, the number of hot objects equal to five, and the experimentally determined optimal configuration of the channels. Objects are numbered in order of decreasing popularity. Results are shown only up to object 100; the curves are relatively flat for objects 101 to 1000.

Recall that the maximum waiting time for requests to a hot object in the static skyscraper system is twice the mean. The dynamic skyscraper system has comparable performance in this respect. Recall from Figure 5 that the maximum waiting time when both hot and cold objects are considered is substantially higher for the static skyscraper/FCFS system than in the system with dynamic skyscraper scheduling for both classes of objects. Thus, the dynamic system yields greater fairness between the two classes of objects, although it still provides better service for the hot objects due to the objective function used in the

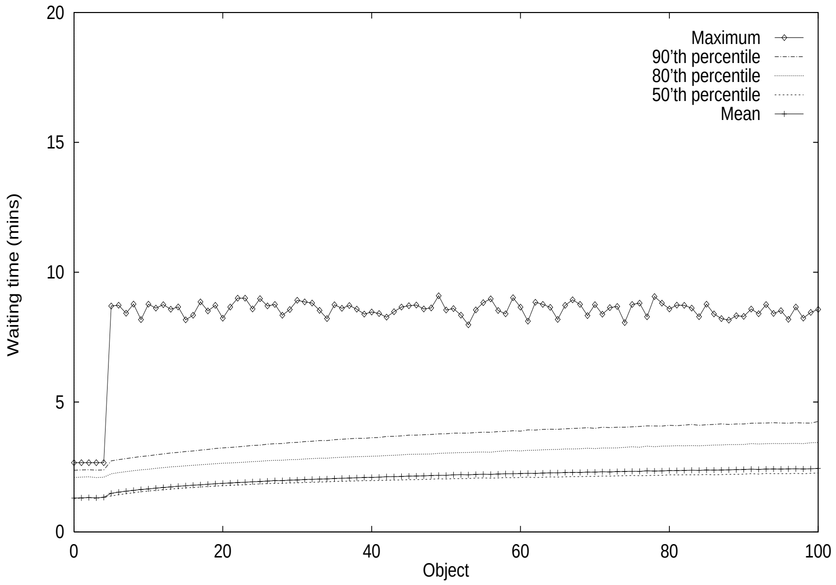

Figure 8: Waiting Time Percentiles vs. Object Popularity.
(1000 120-minute objects; $\lambda = 40$; hot set size = 5;
dynamic skyscraper for hot set with K = 8, W = 16, and 40 channels;
dynamic skyscraper for cold set with K = 3, W = 2, and 1560 channels)

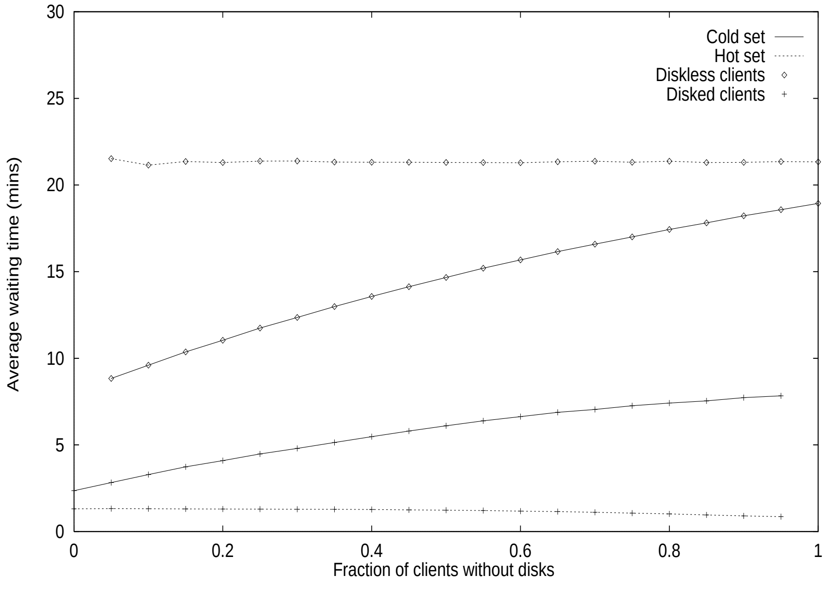

Figure 9: Average Client Waiting Time vs. Fraction of Diskless Clients.
(1000 120-minute objects; $\lambda = 40$; hot set size = 5;
dynamic skyscraper for hot set with K = 8, W = 16, and 40 channels;
dynamic skyscraper for cold set with K = 3, W = 2, and 1560 channels)

channel partitioning. Furthermore, its use of FCFS scheduling of transmission clusters yields reasonable fairness within each class of objects and, considering the inherent randomness of dynamic scheduling, relatively low variance in waiting time for each class. In fact, for the configuration shown in the figure, the ninetieth percentile waiting time is *less than twice the mean* for both the hot and cold object sets.

5.4 Heterogeneous Clients

For both static and dynamic skyscraper systems, the new segment size progressions permit a mix of clients with varying storage and reception capabilities. In Figure 9, we consider a scenario in which a fraction of the clients have very limited local storage, and thus must begin reception at the beginning of a transmission cluster. Dynamic skyscraper scheduling is used for both hot and cold objects, with the number of hot objects equal to five and, as before, the experimentally determined optimal channel configuration.

To conserve channel resources, if a request from a client with limited storage arrives during an active transmission cluster for the requested object, the client can only obtain the next available transmission cluster that starts *after* the point in time at which it is no longer possible for any client to batch in with the existing broadcast. As can be seen in the figure, this policy is reasonably effective in isolating the clients that have buffering capabilities from detrimental performance impact owing to the presence of clients without such capabilities. Furthermore, the differential in performance between the two classes of clients is perhaps reasonable given the differential in cost of the settops. Note that clients are informed of the time at which their broadcast will begin, so viewers with inexpensive settops can plan accordingly.

6 Conclusions

As noted in the introduction, the principal contributions of this paper are:

- a dynamically scheduled skyscraper broadcast scheme that provides client requests with the precise time at which their broadcast will begin, or an upper bound on that time if the delay is small,
- a cost/performance analysis of alternative segment size progressions for the skyscraper scheme,
- new segment size progressions for static or dynamically scheduled skyscraper channels; these progressions improve dynamic scheduling, simplify disk layout, and allow clients with inexpensive settops to receive the skyscraper broadcasts at constrained points in time,

- an analytic formula for first-cut estimates of skyscraper configurations that will have high performance for a given system, and
- a preliminary evaluation of the cost/performance benefit that can be derived from dynamically scheduling the skyscraper channels.

Key observations from the preliminary performance study are that dynamically scheduling the skyscraper channels significantly outperforms static skyscraper broadcasts with respect to overall mean as well as variability in client waiting time. Conversely, the number of channels required for a given target average client waiting time can be significantly lower for the dynamic system. The use of FCFS scheduling of transmission clusters yields reasonable fairness between hot and cold objects and, considering the inherent randomness of dynamic scheduling, relatively low variance in waiting time for each class. In fact, for the representative configuration considered in the preliminary experiments, the ninetieth percentile waiting time is *less than twice the mean* for each object class. We also showed that diskless clients can receive a reasonable level of service without a large negative impact on the performance of clients with more expensive settops.

On-going and future research includes (1) developing a general optimization model for static and dynamic skyscraper configurations and using such configurations to determine more precisely the quantitative benefit of dynamic scheduling over a greater variety of systems, including systems with multiple length objects and a variety of object popularity distributions, (2) evaluating various policies for reassigning unused skyscraper broadcast periods to waiting requests in other transmission clusters, (3) refining the server disk layout strategy for skyscraper broadcasts and generalizing this layout strategy for avoiding start-up latency in conventional broadcasts, (4) investigating the benefit of using extra channels for broadcasting the first unit-segment of each object, to further reduce the delay for batching with an active transmission cluster, and (5) the design of skyscraper broadcast systems with variable bit rate transmissions.

Acknowledgements:
The authors wish to acknowledge Li Fan, Arup Guha, Anirban Mahanti, Dan Sorin, Jayakumar Srinivasan, and Thanos Tsiolis for helpful discussions and comments on this research and for assistance with the analyses contained in this paper.

References

[1] C. C. Aggarwal, J. L. Wolf and P. S. Yu, "A Permutation Based Pyramid Broadcasting Scheme for Video-on-Demand Systems", *Proceeding of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996.

[2] S. Berson, R. Muntz, S. Ghandeharizadeh and X. Ju, "Staggered Striping in Multimedia Information Systems", *Proceedings of the 1994 ACM SIGMOD Conference*, Minneapolis, MN, May 1994, pp. 79-90.

[3] W. J. Bolosky, J. S. Barrera, R. P. Draves, R. P. Fitzgerald, G. A. Gibson, M. B. Jones, S. P. Levi, N. P. Myhrvold and R. F. Rashid, "The Tiger Video Fileserver", *Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'96)*, Zushi, Japan, April 1996.

[4] A. Dan, P. Shahabuddin, D. Sitaram and D. Towsley, "Channel Allocation under Batching and VCR control in Movie-on-Demand Servers", *Journal of Parallel and Distributed Computing 30*, 2 (November 1995), pp. 168-179.

[5] A. Dan and D. Sitaram, "An Online Video Placement Policy based on Bandwidth to Space Ratio", *Proceedings of the 1995 ACM SIGMOD Conference*, San Jose, CA, May 1995, pp. 376-385.

[6] A. Dan, D. Sitaram and P. Shahabuddin, "Scheduling Policies for an On-demand Video Server with Batching", *Proceedings of the ACM Multimedia Conference*, San Francisco, CA, October 1994, pp. 15-23.

[7] S. Ghandeharizadeh and S. H. Kim, "Striping in Multi-disk Video Servers", *Proceedings of the SPIE High-Density Data Recording and Retrieval Technologies Conference*, October 1995.

[8] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems", *Proceedings of the ACM SIGCOMM'97 Conference*, Cannes, France, September 1997, pp. 89-100.

[9] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance*, Prentice Hall, Englewood Cliffs, New Jersey, 1984.

[10] B. Ozden, R. Rastogi, and A. Silberschatz, "Disk Striping in Video Server Environments", *Proceeding of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996, pp. 580-589.

[11] T. S. Perry, "The Trials and Travails of Interactive TV", *IEEE Spectrum 33*, 4 (April 1996), pp. 22-28.

[12] A. K. Tsiolis and M. K. Vernon, "Group-Guaranteed Channel Capacity in Multimedia Storage Servers", *Proceedings of the 1997 ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems*, Seattle, WA, June 1997, pp. 285-297.

[13] S. Viswanathan and T. Imielinski, "Pyramid Broadcasting for Video-on-Demand Service", *Proceedings of the SPIE Multimedia Computing and Networking Conference*, Vol. 2417, San Jose, CA, February 1995, pp. 66-77.

[14] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting", *Multimedia Systems 4*, 4 (August 1996), pp. 197-208.