Dynamic Skyscraper Broadcasts for Video-on-Demand *

Derek L. Eager[1] and Mary K. Vernon[2]

[1] Department of Computer Science
University of Saskatchewan
Saskatoon, SK Canada S7N 5A9
eager@cs.usask.ca
[2] Computer Sciences Department
University of Wisconsin – Madison
Madison, WI 53706
vernon@cs.wisc.edu

Abstract. Skyscraper Broadcasting is a recently proposed statically scheduled broadcast technique for video-on-demand that addresses the network-I/O bottleneck to provide significantly superior performance over previous approaches. This paper defines a scheme for *dynamically* scheduling the objects that are broadcast on the skyscraper channels. 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 not only improve dynamic scheduling, but also simplify the server disk layout problem and allow clients with inexpensive (single-tuner, limited storage) settops to receive skyscraper broadcasts. Preliminary simulation results show that the proposed dynamic scheme (1) provides factors of two or more improvement in mean client waiting time, (2) outperforms the static system with respect to variability in client waiting time, and (3) delivers reasonable service to clients with inexpensive settops while providing clients that have more expensive settops with a high level of service that is relatively isolated from detrimental performance impact from the diskless clients.

1 Introduction

An important goal in the design of video-on-demand systems (e.g., Time Warner's test market system in Orlando [11] or Microsoft's prototype Tiger Video Fileserver [3]) is to reduce the number of channels required to deliver quick response to the end user. To address this issue, 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

* 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.
To appear in the 4th International Workshop on Multimedia Information Systems (MIS '98), Istanbul, Turkey, September 1998.

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 initial 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 system channels are required to provide a given target client response. Of the three schemes that propose and improve on this idea, the segment size progression in the skyscraper broadcast technique offers the lowest latency while requiring a client buffering capability that is easily satisfied by a single commodity disk.

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. 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 investigates several potential improvements in the skyscraper broadcast scheme proposed by Hua and Sheu. 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,
- derive new segment size progressions for the static or dynamic skyscraper channels that simplify the server disk layout problem, allow clients with inexpensive (limited storage) settops to receive the skyscraper broadcasts, and improve dynamic skyscraper system performance, 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 are 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 active skyscraper broadcast, further reducing the response time for the more popular objects.

The remainder of the paper is organized as follows. Section 2 reviews the previously proposed static skyscraper broadcast scheme and defines the new dynamically scheduled alternative evaluated in this paper. Section 3 explores the cost/performance trade-offs of new skyscraper segment size progressions. Simulation results that compare the performance of the static and dynamic skyscraper schemes are presented in section 4, and section 5 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 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.

Table 1. Skyscraper Broadcast System Parameters.

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

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, $\{1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, 105, 105, ...\}$, 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. For example,

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. (This scheme was named skyscraper broadcasting because when the segment sizes are stacked one above the other, they form the shape of a skyscraper.)

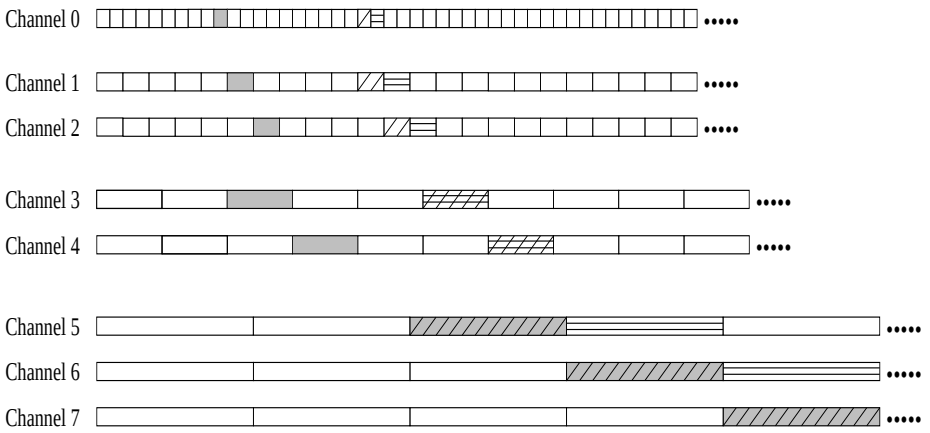

Fig. 1. K Skyscraper Channels for Broadcasting a Single Object. ($K = 8$; $W = 12$)

The gray shading in Figure 1 illustrates the *reception sequence*, or sequence of broadcasts that a client receives, for a client who requests the object just prior to the gray unit-segment broadcast on channel 0. Note that in this reception sequence, 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 during the gap between reception on channels 4 and 5.

Two other reception sequences are shown in the figure: one diagonally striped, the other horizontally striped. Note that these two sequences share broadcasts on channels 3 and 4, while the diagonally striped sequence shares 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 sequence.

Hua and Sheu show that for the given progression 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 storage required 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 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. For example, if K=8, W=12, and T equals 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, if 24 channels are assigned to a two-hour object, a playback will begin every 5 minutes.

2.2 Dynamic Scheduling of Skyscraper Broadcasts

In this paper, rather than devoting K channels to 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. A key question is how to identify reception sequences where it is safe to change the object that is broadcast. Below we identify non-overlapping clusters of skyscraper broadcast periods that can be dynamically scheduled, describe the basic dynamic skyscraper scheme, and an optimization of this scheme termed *channel stealing.*

Skyscraper Transmission Clusters 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 unmarked 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. 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. Note that, according to the above definition for non-overlapping transmission clusters, some broadcast periods such as the diagonally striped segment on channel 0 in Figure 1 are not a member of any transmission cluster, and will not be used in a dynamically scheduled system that has the segment size progression defined by Hua and Sheu. We address this issue in section 3.

The Basic Dynamic Scheme Let C channels in the system be allocated to the dynamically scheduled skyscraper broadcasts for objects of a given playback length and rate, and let these channels be organized into N groups of K channels each. C and K are parameters of the configuration. As in the static skyscraper system, each group of K channels has a fixed segment size progression that is upper bounded by the parameter W. For now the reader should assume that

the segment size progression is the same as defined by Hua and Sheu 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.

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 4. 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. This policy turns out to provide noticeably improved performance when temporary channel reassignment is implemented.

It is also possible to temporarily 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.

3 Alternative Segment Size Progressions

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 dynamic skyscraper system because particular segment broadcasts can't be used in any transmission cluster.

To address the problem of unused channel bandwidth, we investigate the cost/performance implications of alternative segment size progressions. As will be discussed below, new segment size progressions not only allow all broadcast periods to be used in the dynamic system, but have two further advantages for static as well as dynamic skyscraper systems: (1) they provide service for clients that have settops with a single tuner and very limited storage capacity, and (2) they simplify the server disk layout problem.

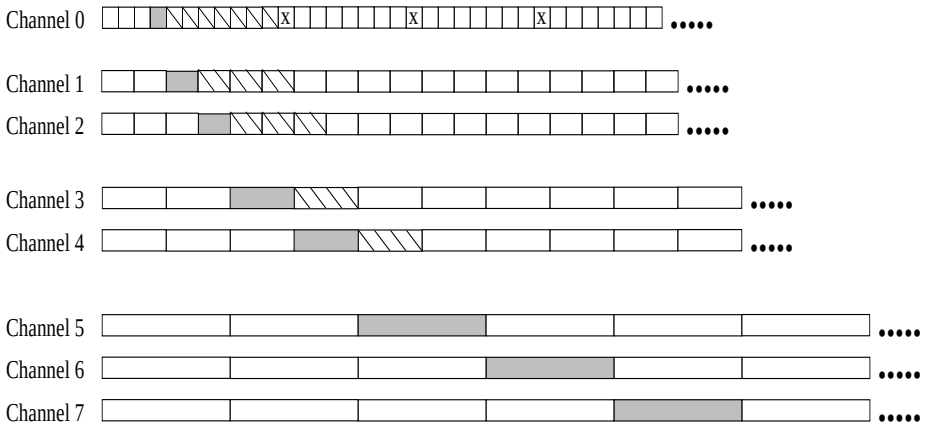

Fig. 2. Segment Size Progression $A = \{1, 2, 2, 4, 4, 8, 8, 16, 16, ...\}$. ($K = 8$; $W = 8$)

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. 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 $C = \{1, 2, 2, 6, 6, 12, 12, 36, 36, ...\}$. Progression A is 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
- 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 first reception sequence in 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,
- due to condition three, 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 broadcast on channel i or the end of the broadcast 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 broadcast on channel i, or the end of the broadcast 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. 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 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.

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

require reception on up to three channels. Progression A has somewhat smaller segment size increases than the progression defined by Hua and Sheu and thus will have slightly higher expected waiting time in a static skyscraper system. 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 somewhat 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 that can still be easily accommodated by a single commodity disk) in each client settop. Furthermore, each of the new progressions can provide (a lower level of) service to clients with very inexpensive single-tuner settops that can only buffer the frame that is currently being received, by serving requests from such clients in the first reception sequence of a new transmission cluster. This capability is evaluated further in section 4.

The new segment size progressions also simplify the server disk layout problem. A server disk layout that makes dynamic scheduling of the skyscraper channels feasible is omitted due to space constraints, but involves adjusting the optimal stripe unit size so that one transmission cluster of data for each segment of a given object is striped across the available disks an integral number of times.

4 Experimental Results

Models for computing optimal dynamic and static skyscraper system configurations, which would support a fairly complete comparison of these systems over a complex design space, are beyond the scope of this paper. Instead, this section provides some preliminary simulation results to illustrate the potential of the dynamically scheduled skyscraper system. In particular, 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. We focus on an 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 provided in Figures 3 and 5 for average client waiting time have 95% confidence intervals that are within 10% of the reported values.

4.1 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.[1] 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, under the following constraints. 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 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. [2]

Figure 3 gives the average client waiting time in each of the three systems, for a system with 1000 objects each with a 120 minute playback duration and Zipf(0) selection frequencies, 1600 total channels, and a client request arrival rate of 40 per minute. For small hot set sizes the overall average client waiting time is very similar in the static skyscraper/FCFS system and the dynamic skyscraper/FCFS system. As the hot set size increases, however, the performance of the static skyscraper/FCFS combination begins to deteriorate, whereas the best performance with the dynamic skyscraper/FCFS combination is achieved in the limiting case when all channels are used for dynamic skyscraper scheduling of 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; at hot set sizes of five to fifteen, the average client waiting time is about two thirds of the lowest average waiting time in the dynamic skyscraper/FCFS combination.

Figure 4 shows that the use of dynamic skyscraper scheduling for both the hot and cold sets also yields improved performance with respect to the maximum

[1] 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].

[2] Note that using the sum of mean and maximum waiting time as 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.

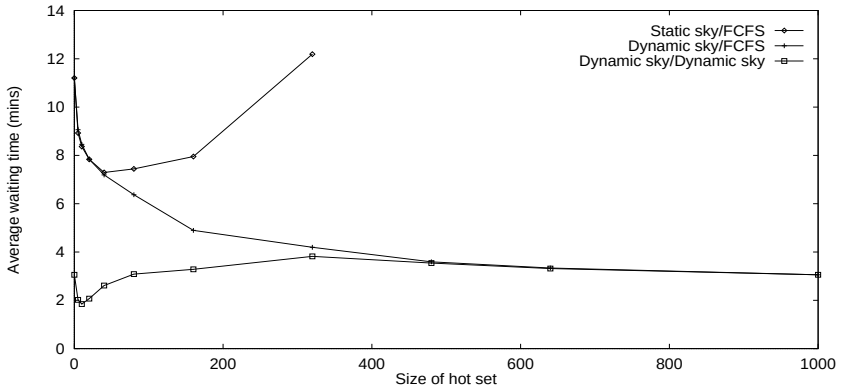

Fig. 3. Average Client Waiting Time vs. Hot Set Size.
(1000 120-minute objects; 1600 channels; $\lambda = 40$)

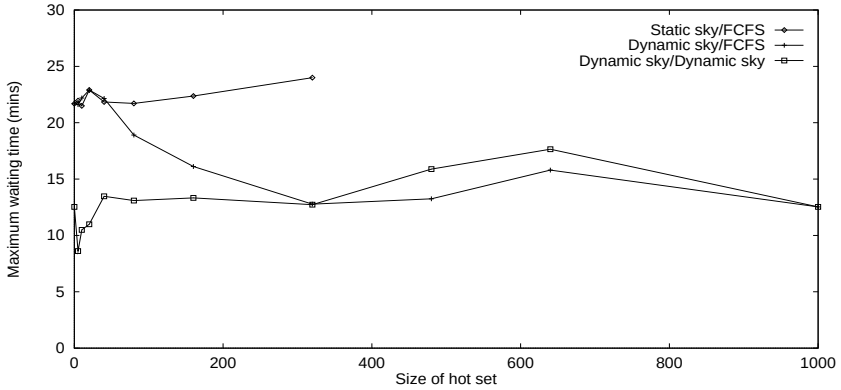

Fig. 4. Maximum Client Waiting Time vs. Hot Set Size.
(1000 120-minute objects; 1600 channels; $\lambda = 40$)

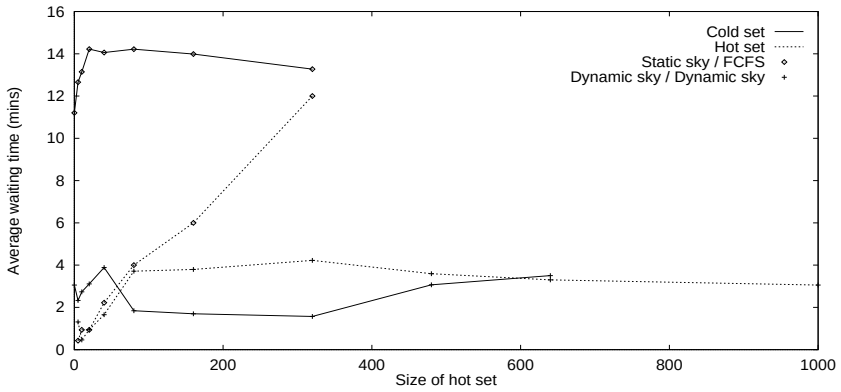

Fig. 5. Average Waiting Time For Hot and Cold Objects vs. Hot Set Size.
(1000 120-minute objects; 1600 channels; $\lambda = 40$)

waiting time observed in the simulations. With a hot set size of five, 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 in the static skyscraper/FCFS system, and is again about two thirds of the lowest maximum waiting time for the dynamic skyscraper/FCFS system.

Figure 5 gives the average client waiting time for each class of 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 use of dynamic skyscraper scheduling on both the hot and cold sets can offer substantially improved fairness in mean waiting time between the two classes of objects as compared to the static skyscraper/FCFS combination.

4.2 Variability in Client Waiting Time

Measures of the distribution of waiting time for each object, omitted due to space constraints, show the following for the system with dynamic skyscraper broadcasts for both hot and cold object sets, hot set size equal to five, and the experimentally determined optimal configuration of the channels: (1) maximum waiting time for each hot object is roughly twice the mean waiting time for the object (as in the static skyscraper/FCFS system), (2) maximum waiting time for each cold object is significantly (approximately three times) higher than the maximum observed waiting time for each hot object due to the objective function used in the channel partitioning (yet Figure 4 shows that the maximum waiting time for cold objects is still significantly better than in the static skyscraper/FCFS system), (3) the 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 evaluated, the ninetieth percentile waiting time is *less than twice the mean* for each object, hot or cold.

4.3 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 this section, 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. 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. Results that are omitted due to space constraints show that this policy is effective in isolating the clients that have settop buffering capabilities from detrimental performance impact owing to the presence of clients

without such capabilities. In particular, mean wait for the hot (or cold) objects by clients with settops that have the buffering capability is non-increasing (or slowly increasing) as the fraction of diskless clients increases. Furthermore, the five to ten-fold 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.

5 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,
- 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,
- 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 dynamic skyscraper systems can significantly outperform static skyscraper systems 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 find that diskless clients can receive a reasonable level of service without a large negative impact on the performance of clients with more expensive settops.

Our on-going and future research plans include (1) developing optimization models for static and dynamic skyscraper configurations and using such configurations to determine the 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) more detailed exploration of server disk layout strategies for skyscraper broadcasts, (4) the design of skyscraper broadcast systems for variable bit rate transmissions, and (5) the caching of skyscraper segments in widely-distributed VOD servers.

Acknowledgements

The authors wish to acknowledge Li Fan, Arup Guha, Anirban Mahanti, Dan Sorin, Jayakumar Srinivasan, and Thanos Tsiolis for assistance with our simulations and for helpful discussions and comments on this research.

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.