Workload-Aware Resource Sharing and Cache Management for Scalable Video Streaming

Bashar Qudah Nabil J. Sarhan
The Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI, 48202 USA
e-mail: {bqudah,nabil}@wayne.edu

October 22, 2007

Part of the paper was presented at MASCOTS 2006.
This work is supported in part by NSF grant CNS-0626861.

Abstract

The required real-time and high-rate transfers for multimedia data severely limit the number of requests that can be serviced by *Video-on-Demand* (VOD) servers. Resource sharing techniques can be used to address this problem. We evaluate through extensive simulation major resource sharing techniques from different classes, considering both the *True Video-on-Demand* (TVOD) and *Near Video-on-Demand* (NVOD) service models. The TVOD model helps in analyzing the server communication bandwidth and disk I/O bandwidth requirements, whereas the NVOD model helps in comparing the achieved throughput, waiting times, and unfairness towards unpopular videos. We utilize this extensive analysis in developing a workload aware hybrid solution (WAHS) that combines the advantages of the best among resource sharing techniques. Moreover, we propose a statistical cache management approach (SCM) and derive analytical models for optimal cache allocation to reduce the demands on the disk I/O when various resource sharing techniques are used.

0.1 Introduction

Video-on-Demand (VOD), though has been around for a while as an alternative delivery method, is advancing slowly but surely towards a more market share, and is expected to exceed that of broadcast-based TV programming and DVD movie rentals at stores. The main driving forces behind this phenomena are the increasing expectations of viewers as new generations enter the market, and the industry interest in more control and lower cost of the delivery process. However, the pace of this advancement is decided by the quality and speed of solutions in the face of the main challenges.

In the technical side, the challenges are greater and harder to solve, the more popular VOD becomes. The number of concurrent video streams that can be supported by a VOD server is highly constrained by the required real-time and high-rate transfers. These requirements quickly consume server resources, including network bandwidth and disk I/O bandwidth.

Resource sharing techniques face this challenge by utilizing the multicast facility. Two resource sharing paradigms dominate scalable video streaming; *stream merging* [21, 5, 11, 29, 3, 27, 28] and *periodic broadcasting* [14, 20, 22, 24, 23, 2, 33, 32]. Stream merging techniques combine streams when possible to reduce the delivery costs. These techniques include *Patching* [6, 21], *Transition Patching* [5, 7], and *Earliest Reachable Merge Target (ERMT)* [11, 13]. Periodic broadcasting techniques, such as *Skyscraper Broadcasting* (SB) [22], *Greedy Disk-conserving Broadcasting* (GDB) [14], *Harmonic Broadcasting* (HB) [24], and *Fibonacci Broadcasting* (FB) [20], divide each supported video into multiple segments and broadcast them periodically on dedicated channels. While stream merging suits wider spectrum of video workloads, periodic broadcasting can be more efficient in serving the extremely popular videos. The decision which paradigm to follow and which technique to use has a great impact on the system overall performance and throughput, and on the quality of service perceived by clients.

Though efforts have been spent on designing and optimizing techniques in each group, the option of utilizing both at the same time was not considered beyond a casual mentioning, either to justify the need for another periodic broadcasting technique, or at best arbitrarily by serving some popular videos in the system by a periodic broadcasting technique without a clear criteria on how to choose them. *Selective Catching* [16] (and *Catching* [16]) is an exception to this general pattern, but it suffers from two main problems. First, as we demonstrate in this paper, the chosen techniques to mix are not the

best each group can offer. Second, modifying a periodic broadcasting technique to fit a stream merging like behavior causes the periodic broadcasting technique to lose its source of strength.

To start with, the literature lacks detailed comparative analysis of various resource sharing strategies. With the many available resource sharing techniques from different classes, it is unclear which one is the best to use in a target environment. For example, it is unclear how ERMT performs in comparison with Selective Catching and Transition Patching and how periodic broadcasting techniques perform compared with stream merging techniques. Moreover, only limited performance metrics, workloads, and service models were considered.

Furthermore, there is very little work on cache management for reducing the demands on the disk I/O when resource sharing techniques are applied. This is, however, an important issue for designing cost effective and scalable VOD servers, especially because of the widening gap between technology improvements in the capacities and speeds of hard disk drives [18]. In [4], the impact of caching was reported by actual measurements for only limited caching schemes and limited and simple resource sharing techniques. The results show that without caching, the disk I/O bandwidth may become a bottleneck.

This paper addresses those limitations. The main contributions of this study can be summarized as follows.

- We evaluate through extensive simulation major resource sharing techniques under both the *True Video-on-Demand* (TVOD) and *Near Video-on-Demand* (NVOD) service models and for two target workloads, one consisting of videos with varying popularities and the other consisting of only popular videos.

- We propose a new *workload aware hybrid solution* (WAHS) that combines the advantages of the best in stream merging and periodic broadcasting techniques. The solution is armed with a dynamic configuration algorithm capable of responding to any significant change in the workload or the system available resources, and it is tuned for high server throughput.

- We propose a *statistical cache management approach* (SCM). With this approach, the server periodically computes video access frequencies and determines the data to be cached based on these statistics. Besides

being performance effective, this approach is easy to implement and incurs small overhead as updates are triggered only when the workload varies considerably. We derive analytical models for allocating cache space for various resource sharing techniques. This work differs significantly from previous work on proxy caching [25] (and references within), which has different objectives and factors affecting the allocation decisions. In addition, our approach differs from low-level caching techniques, such as *Least Recently Used* (LRU), *Least Frequently Used* (LFU), and *Interval Caching* [8, 30], whereby the content of the cache changes frequently, incurring significant overheads. SCM is compared to *Interval Caching* [9] and is shown to achieve significantly better performance when implemented with the most aggressive resource sharing techniques.

- We develop analytical models for optimal tuning of design parameters for Transition Patching and Selective Catching.

- We introduce the load on the underlying storage subsystem as an additional dimension to the comparisons among various resource sharing techniques and examine the impact of caching on reducing these requirements.

The performance evaluation considers four target environments: *mixed-video workload and TVOD*, *hot-video workload and TVOD*, *mixed-video workload and NVOD*, and *hot-video workload and NVOD*. While the mixed-video environments are more realistic, the hot-video environments are used to analyze periodic broadcasting that are designed for popular videos. In TVOD, we compare the server resources required to service all requests immediately. In NVOD, however, we fix server resources and compare the number of customers that can be serviced concurrently, the waiting times, and unfairness towards unpopular videos. This study examines the impacts of many system and workload parameters, including customer waiting tolerance, request arrival rate, number of videos, server network bandwidth, and cache size. The effect of request scheduling is also discussed. Furthermore, we introduce and apply a framework for comparing periodic broadcasting techniques with other techniques. We consider here the ability of periodic broadcasting techniques to limit the maximum waiting time and to provide waiting time guarantees.

The rest of this paper is organized as follows. Section 0.2 discusses major resource sharing techniques and cache management. WAHS, the proposed hybrid solution is presented in Section 0.3, and SCM, the proposed cache management approach is presented in Section 0.4. Section 0.5 shows how design parameters can be tuned optimally. Section 0.6 discusses the performance evaluation methodology and workload characteristics and presents the main results. Finally, conclusions are drawn in the last section.

0.2 Background Information

0.2.1 Resource sharing

Let us now discuss the main resource sharing techniques analyzed in the paper: Patching, Transition Patching, ERMT, Selective Catching, FB, SB, GDB, and HB. This study excludes techniques that achieve low resource sharing (such as Batching[10]) or reduce playback quality (such as Piggybacking [17]). To conduct fair comparisons and account for limitations in client bandwidth, we do not consider techniques that require client download bandwidth more than double the video playback rate, except for Harmonic Broadcasting (HB). Despite requiring high download bandwidth in the client, HB is very effective in reducing the required server bandwidth. In addition, we exclude techniques that are designed for heterogeneous receivers such as *Enhanced Hybrid Solution (EHS)* [27], *HeRo* [2], *BroadCatch* [33], and *Optimized Heterogeneous Periodic Broadcasting (OHPB)* [32] for losing their advantages in homogeneous environments and performing similar or worse than their counterpart techniques that are designed for homogeneity.

Stream Merging Techniques

While Batching services all waiting requests for a video using one full multicast video stream and requires only one client download channel at the video playback rate. Patching expands the multicast tree dynamically to include new requests. A new request joins the latest *regular* (i.e., full) stream for the object and receives the missing portion as a *patch*. Hence, it requires two download channels (each at the video playback rate) and additional client buffer space. When the playback of the patch is completed, the client continues the playback of the remaining portion using the data received from the

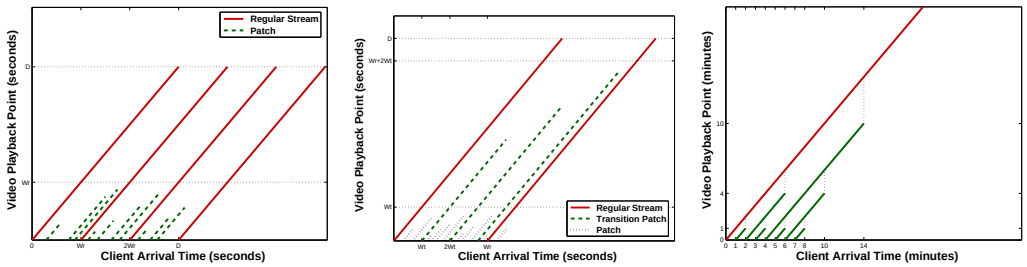
multicast stream and already buffered locally. To avoid the continuously increasing patch lengths, regular streams are retransmitted when the required patch length exceeds a pre-specified value called *regular window* (Wr). Figure 1(a) further explains the concept in servicing one video of D seconds in length. (Typically, Wr is much smaller than D.)

Transition Patching allows some patches to be sharable by extending their lengths. It introduces another multicast stream, called *transition patch*. The threshold to start a regular stream is Wr as in Patching, and the threshold to start a transition patch is called the *transition window* (Wt). The transition patch length is equal to the difference between the starting times of the transition patch and the last regular stream plus $2Wt$, whereas the length of the patch is equal to the difference between the starting times of the patch and the latest transition stream or regular stream. Hence, the maximum possible patch length is Wt, and the maximum possible transition patch length is $Wr + 2Wt$. Figure 1(b) further illustrates the concept. A possible scenario for a client is to start listening to its own patch and the transition patch, and when its patch is completed, it starts listening to the regular stream.

ERMT is a near optimal hierarchical stream merging technique. It also requires two download channels (each at the playback rate), but it makes each stream sharable and thus leads to a dynamic merge tree. A new client joins the closest reachable stream (*target*) and receives the missing portion by a new stream (*merger*). Reachable means the merge can occur before the target terminates. After the merger stream finishes and merges into the target, the later can get extended to satisfy the playback requirement of the new client(s), and this extension can affect its own merge target. For example, in Figure 1(c), the third stream length got extended from two minutes to four minutes after the fourth stream had merged with it. ERMT is used by *Bandwidth Skimming* [12] to work when the client download bandwidth is less than double the playback rate.

Periodic Broadcasting Techniques

SB, GDB, and FB divide each video into a number of non-equal segments and broadcast each segment periodically on a dedicated channel at the video playback rate. The client waits until the beginning of the next broadcast of the first segment and then receives data concurrently from two broadcast channels. The relative length of the n^{th} segment compared to the first seg-



(a) Patching (b) Transition Patching (c) ERMT

Figure 1: Stream Merging Techniques

ment is determined using a partitioning series. Each protocol has a different series as shown in Table 1.

Table 1: Segment Partitioning in Periodic Broadcasting

Technique	**Partitioning Series**
SB	$1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, 105, 105, 212, ...$
FB	$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ...$
GDB	$1, 2, 2, 5, 5, 12, 12, 25, 25, 60, 60, 125, 125, 300, ...$
Modified GDB	$1, 1, 1, 2, 2, 5, 5, 12, 12, 25, 25, 60, 60, 125, 125, ...$

On the other hand, HB divides each video into a number of equal segments and broadcasts them periodically on channels with decreasing bandwidth. The bandwidth for channel i is b/i, where b is the playback rate. The client has to receive data concurrently from all broadcast channels allocated for the requested video. HB has a continuity problem, which was resolved by Cautious Harmonic Broadcasting (CHB) and Quasi Harmonic Broadcasting (QHB) [26]. Both still require clients to receive data from all broadcast channels.

Composite Techniques

Catching uses a modified version of GDB (whose series is shown in Table 1) to deliver the hot videos, but instead of making the client wait until the beginning of the next broadcast time, the client receives the small missed portion by a patch. A client initially listens to one broadcast channel and its own patch, and then to two broadcast channels. Selective Catching extends

Catching by delivering the cold videos using *Controlled Multicast* [15], which works essentially the same as Patching.

0.2.2 Cache Management

Interval caching exploits the high skewness in video access patterns by caching intervals between successive streams accessing the same contents in the main memory of the server. Specifically, interval caching reuses the data brought by a stream in servicing a closely following stream if sufficient cache space exists to hold the data blocks in the cache. The two streams are called the *following* stream and the *preceding* stream, respectively. The following stream will have to be serviced from disks until the playback reaches the first cached block. The required cache space for servicing the following stream depends on the time interval between the two streams and the compression method used. Interval caching uses the cache (memory) space efficiently by victimizing longer intervals for caching shorter ones.

0.3 Workload Aware Hybrid Solution (WAHS)

Two resource sharing paradigms dominate scalable video streaming; stream merging and periodic broadcasting. While the former suits wider spectrum of video workloads, the later can be more efficient in serving extremely popular videos. The decision on which paradigm to follow and which technique to use has a great impact on the system overall performance and throughput, and on the quality of service perceived by clients.

Though efforts have been spent on designing and optimizing techniques in each group, the option of utilizing both at the same time was not considered beyond a casual mentioning, either to justify the need for another periodic broadcasting technique, or at best arbitrarily by serving some popular videos in the system by a periodic broadcasting technique without a clear criteria on how to choose them. *Selective Catching* [16] (and *Catching* [16]) is an exception to this general pattern, but it suffers from two main problems. First, as we demonstrate in this paper, the chosen techniques to mix are not the best each group can offer. Second, modifying a periodic broadcasting technique to fit a stream merging like behavior causes the periodic broadcasting technique to lose its source of strength. As we show later in this paper, Selective Catching performs worse than some of the efficient stream merging

techniques without even adding any periodic broadcasting componenet to them.

We propose a workload aware hybrid solution (WAHS) that provides clear approaches for answering all key configuration questions. The configuration of the system is not static but dynamic and responds to any significant change in the workload or the available system resources. The solution combines the best in each group (as will be shown in Section 0.6), ERMT as the stream merging technique with Fibonacci periodic broadcasting. Fibonacci is used for the most popular videos while ERMT is used for the remaining. However the Hybrid solution might dynamically converge under certain conditions to one of the two techniques when that potentially achieves better performance. Such as a combination of very low request rate and limited server resources might lead to pure ERMT. But in general, the decision is more involving and needs to take into consideration several factors at once and to response to changes dynamically, as well. The main three configuration parameters the approach tries to set optimally are: (1) How many videos to serve using each technique? (2) How to split the available resources between the two groups of service? (3) And how to distribute the resources reserved for the broadcasting group among the broadcasting videos for best possible performance? Within the ERMT group the resources are shared by requests of all videos in the group and managed by the scheduling policy in place. The hybrid solution is primarly tuned by the configuration process for higher throughput then for lower average customer waiting time. However the order can be easily interchanged if desired. Another issue, while system resources include more than server network channels Ch_{Ava}, such as the I/O channels and the cache space, for their exceptional importance and simplicity, server network channels are the primary focus in the discussion below. However, other resources can be included in the decision process as needed.

0.3.1 Step I: Forming The Two Groups

The first step in this direction is deciding whether forming two groups is a better choice. Given the expected workload, customer waiting and defection behavior, and the level of resources the system does allocate for service, can ERMT provide a TVOD service on its own? If yes, problem is solved and no need for hybridizing till conditions are changed. However, if the answer is no, another question needs to be answered. Is Fibonacci capable of providing a zero-defection (0-D) service by itself? If answer is yes, again problem is

Table 2: Symbols used in algorithm description

Parameter Name	**Symbol**
Number of Videos Served using Fibonacci, ERMT	V_{FB}, V_{ERMT}
Total Available Server Channels	Ch_{Ava}
ERMT Requirement to provide a TVOD service for Videos s till t, for the i_{th} Video	$Ch_{ERMT:TVOD_{s:t}}$, $Ch_{ERMT:TVOD_i}$
Fibonacci Requirement to provide a Zero-defection service for Videos s till t, for the i_{th} Video	$Ch_{FB:Zero_{s:t}}$, $Ch_{FB:Zero_i}$
Channels Allocated for All Fibonacci Videos, for the i_{th} Video	Ch_{FB}, Ch_{FB_i}
Channels Shared by All ERMT Videos	Ch_{ERMT}
Request Arrival Rate for (Fibonacci, ERMT) Videos (Request/s)	λ_{FB}, λ_{ERMT}
Predicted Defection Probability for All Videos, for the i_{th} Video	DP, DP_i
Predicted Defection Probability for (Fibonacci,ERMT) Videos	DP_{FB}, DP_{ERMT}
Predicted Average Customer Waiting Time for All Videos, for the i_{th} Video	$ACWT$, $ACWT_i$
Predicted Average Customer Waiting Time for (Fibonacci,ERMT) Videos	$ACWT_{FB}$, $ACWT_{ERMT}$

solved and no need for hybridizing till conditions are changed. Unfortunately, the answer for both questions in many practical systems is no. In this case, the coexistence solution has to be considered and tested. The approach we propose is comparing each video's ERMT TVOD channel requirement with its Fibonacci 0-D requirement, and placing the video in the group that requires the least. That does not mean the video is guaranteed that smaller requirement. just it is better to serve the video using the technique requiring the least regardless of the actual resource allocation for the video. Assuming homogeneity in customer behavior across all videos, that is the customer behavior is not a function of the requested video, but a function of other service parameters including service paradigm and technique, which is typically the case, the 0-D requirement is the same for all videos. However, ERMT requirement remains different for different videos as a function of video request rate, and it is higher the more popular the video is.

Predicting ERMT TVOD Requirement

The ERMT TVOD requirement to serve all V videos can be predicted using the following:

$$Ch_{ERMT:TVOD_{1:V}} = \sum_{i=1}^{V} Ch_{ERMT:TVOD_i}. \quad (1)$$

Since all videos actually share the pool of resources in case of ERMT, therefore, when the number of videos are large enough (which is typically the case), $Ch_{ERMT:TVOD_i}$ can be substitute with the average server channel requirement for the i_{th} video (versus the actual or maximum requirement), which can be predicted using [12]:

$$Ch_{ERMT:TVOD_i} = \eta * \log(\lambda_i \times D_i/\eta), \quad (2)$$

where λ_i is the request arrival rate (request/second), and D_i is the duration in seconds of the i_{th} video. η found by [12] to be 1.62 for an optimal stream merging technique requiring two channels in the client side. However, since ERMT is not the optimal off-line technique $\eta = 1.62$ was derived for, but it is the best known practical on-line stream merging technique, a higher value of η is required. As presented in Figure 2(a), we found experimentally that $\eta = 2.0$ works better for ERMT.

Predicting Fibonacci 0-D Requirement

Fibonacci 0-D channel requirement per video is independent from video request rate, but depends solely on customer defection behavior. What is required is a cap on the maximum waiting time (first segment length) below the customer waiting tolerance that with time of service guarantee, which Fibonacci is capable of providing, results in almost zero-defection. Once that value is known, the required number of channels are easy to calculate, to result in a first segment of shorter length.

Table 3: WAHS Top Level Description

Step	Description
1	**Select Videos to be Served using Fibonacci Periodic Broadcasting (V_{FB})**
Continued on next page	

Table 3 – continued from previous page

Step	Description
1.1	Estimate ERMT Channel Requirement to provide a TVOD service ($Ch_{ERMT:TVOD_{1:V}}$) $V_{FB} = 0$; $\eta = 2.0$; $Ch_{ERMT:TVOD_{1:V}} = \sum_{i=1}^{V} \eta * log(N_i/\eta)$; *if* ($Ch_{ERMT:TVOD_{1:V}} < Ch_{Ava}$) *proceed to Step* 2;
1.2	Estimate Fibonacci Broadcasting Channel Requirement to provide a Zero-defection service. ($Ch_{FB:Zero_{1:V}}$) *if* ($Ch_{FB:Zero_{1:V}} < Ch_{Ava}$) { $V_{FB} = V$; *proceed to Step* 2; }
1.3	Estimate for each Video (i) both $Ch_{ERMT:TVOD_i}$ and $Ch_{FB:Zero_i}$ *for* ($i = 1$, $i <= V$, $i++$) { *if* ($Ch_{FB:Zero_i} < Ch_{ERMT:TVOD_i}$) {$V_{FB}++$;} *else* {*break*;} }
2	**Allocate Channels for Broadcasting (Ch_{FB})**
2.1	Minimize Total Customer Defection Probability (DP) $Ch_{FB} = minimum(Ch \times \frac{\lambda_{FB}}{\lambda}, Ch_{FB:Zero_{1:V_{FB}}})$; $Ch_{ERMT} = Ch - Ch_{FB}$; *while* ($DP_{FB} > 0.0$ & $Ch_{ERMT} > 0$ & $DP < currentDP$) { $currentDP = DP$; $Ch_{FB}++$; $Ch_{ERMT}--$; *re-predict:* DP_{FB}, DP_{ERMT}, *and* DP }
2.2	Minimize Total Customer Waiting Time ($ACWT$) *while* ($DP == 0.0$ & $Ch_{ERMT} > 0$ & $ACWT < currentACWT$) { $currentACWT = ACWT$; $Ch_{FB}++$; $Ch_{ERMT}--$; *re-predict:* $ACWT_{FB}$, $ACWT_{ERMT}$, *and* $ACWT$ }
3	**Distribute Broadcast Channels among Broadcast Videos**

Continued on next page

Table 3 – continued from previous page

Step	Description
3.1	Allocate minimum channels per video (*if possible, round robin, from most popular to least*)

3.2	Minimize Total Customer Defection Probability (DP)
	while ($DP_{FB} > 0.0$ & $allocatedChannels < Ch_{FB}$) {
	/* *Pick the video to get the next channel* */
	$maxGain = 0.0$; $pickedVideo = 1$;
	for ($i = 1$, $i <= V_{FB}$, $i++$) {
	$gain = (DP_i(Ch_{FB_i}) - DP_i(Ch_{FB_i} + 1)) \times \lambda_i$;
	if ($gain > maxGain$) {
	$maxGain = gain$; $pickedVideo = i$; } }
	$Ch_{FB_{pickedVideo}}++$; $allocatedChannels++$; }

3.3	Minimize Total Customer Waiting Time ($ACWT$)
	while ($allocatedCh < Ch_{FB}$) {
	/* *Pick the video to get the next channel* */
	$maxGain = 0.0$; $pickedVideo = 1$;
	for ($i = 1$, $i <= V_{FB}$, $i++$) {
	$gain = (ACWT_i(Ch_i) - ACWT_i(Ch_{FB_i} + 1)) \times \lambda_i$;
	if ($gain > maxGain$) {
	$maxGain = gain$; $pickedVideo = i$; } }
	$Ch_{FB_{pickedVideo}}++$; $allocatedChannels++$; }

0.3.2 Step II: Splitting The Resources

The next step is the optimal allocation of the available resources across the two groups. The allocation should reflect the importance of the different objectives of the system; a higher throughput (lower defection probability), followed by a lower average customer waiting time. This requires fairly accurate prediction for these two metrics in case of both ERMT and Fibonacci. Having such prediction, the two metrics can be evaluated for all allocation options and the one achieves the lowest defection probability is the choice. Ties can be broken based on the expected average customer waiting times.

The overall predicted defection probability (DP) and average customer waiting time ($ACWT$) can be calculated by the weighted averages of the corresponding group level metrics:

$$DP = DP_{FB} \times \frac{\lambda_{FB}}{\lambda} + DP_{ERMT} \times \frac{\lambda_{ERMT}}{\lambda}, \tag{3}$$

and

$$ACWT = ACWT_{FB} \times \frac{\lambda_{FB}}{\lambda} + ACWT_{ERMT} \times \frac{\lambda_{ERMT}}{\lambda}. \tag{4}$$

Predicting Fibonacci Metrics: DP_{FB} and $ACWT_{FB}$

Fibonacci group level metrics; DP_{FB} and $ACWT_{FB}$, can be calculated in their turn by the weighted averages of the corresponding video level metrics::

$$DP_{FB} = \sum_{i=1}^{V_{FB}} DP_i \times \frac{\lambda_i}{\lambda_{FB}} \tag{5}$$

and

$$ACWT_{FB} = \sum_{i=1}^{V_{FB}} ACWT_i \times \frac{\lambda_i}{\lambda_{FB}}. \tag{6}$$

The real challenge is in predicting the video level metrics. The video level defection probability, in case of Fibonacci, is a function of the customer waiting and defection model and the channels allocated for the video. The first segment length (l_i) can be easily calculated based on the available channels, which represents the maximum waiting time for any customer requesting that particular video. Assuming equal probability for request arrival during that period (which is typically usually the case) Then DP_i can be calculated by:

$$DP_i = \int_0^{l_i} F(x).dx, \tag{7}$$

where $F(x)$ is the cumulative distribution function of the customer waiting tolerence model (i.e. $F(x)$ is the probability the customer maximum waiting time is $\leq x$). The customer waiting tolerence model can be built statistically based on system historical information. However, the video allocated channels is not known yet. The number of videos in the group is known from Step I, and the above sub-process is executed for an assumed group channel allocation, therefore, distributing the examined group channel allocation

among the group videos is required to know the assumed video allocated channels. One trivial solution is to distribute the channels equally among the group videos, however a more optimal solution is explained in Step III. The distribution algorithm need to be executed iteratively in Step II before it is finally executed for the actual group allocation; the output of Step 2. The video level average customer waiting time is much easier and a function of only the video channel allocation. Simply, $ACWT_i = l_i/2$. Figure 2(b) demonstrates how accurate the predition of the defection rate is in case of Fibonacci.

Predicting ERMT Metrics: DP_{ERMT} and $ACWT_{ERMT}$

Unlike Fibonacci, the ERMT group level metrics are harder to predict. The resource pooling, scheduling policy interaction, the dependency on the request rate, and the dynamic nature of the technique, all make prediction complex and hard with stream merging techniques, and ERMT is no exception. Though ERMT does not isolate videos during service, but to simplify this complexity, we follow the divide and conquer approach by performing the prediction at the video level, as we did with Fibonacci. Therefore, as in Fibonacci, the group level metrics; DP_{ERMT} and $ACWT_{ERMT}$, are calculated in their turn by weighted averages of the corresponding video level metrics::

$$DP_{ERMT} = \sum_{i=V_{FB}+1}^{V} DP_i \times \frac{\lambda_i}{\lambda_{ERMT}} \tag{8}$$

and

$$ACWT_{ERMT} = \sum_{i=V_{FB}+1}^{V} ACWT_i \times \frac{\lambda_i}{\lambda_{ERMT}}. \tag{9}$$

However a series of modifications are introduced to fit that approach for the ERMT case.

The *first* modification is in predicting the video actual average share of channels. Though the resources are pooled with ERMT, but practically at a snapshot of time the channels has to be allocated for the different video requests managed by the scheduling policy. We try to predict the average share over long period of time, but that lack the accuracy in case of Fibonacci, where the allocation is static and known in advance. Under a fair scheduling policy like *FCFS*, each video is expected to receive in average a share of

channels as a function of its request rate compared the request rates of the other videos sharing the resources. The predicted average share of the i_{th} video under such a fair policy can be calculated using:

$$Ch_i = \frac{\log(\lambda_i \times D_i)}{\sum_{v=V_{ERMT}}^{V} \log(\lambda_v \times D_v)} \times Ch_{ERMT}. \tag{10}$$

The log represent the relation between the required channels and the request rate in case of ERMT as shown in Equation 2. However, under more biased policies like MQL, popular videos can receive more than their fair share.

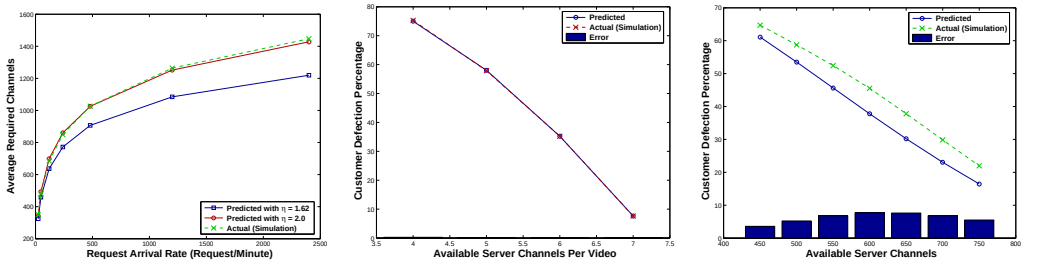
The *second* modification is predicting the average video service period. Again, unlike Fibonacci, where the period is accurate, repeatable, and guaranteed, with ERMT it is just a prediction for the average over long period. Given the predicted channels for a given video (i), the expected maximum request arrival rate that can be supported for a TVOD service using ERMT can be calculated using the the reverse of Equation 2, which represents the expected average value for the stream initiation rate for the video (λ_{S_i}). The inverse of the rate is the expected average period between the consecutive streams for the video (l_i). Same as with Fibonacci, $DP_i = \int_0^{l_i} F(x).dx$ and $ACWT_i = l_i/2$. However, the customer waiting tolerence model ($F(x)$) with ERMT very likely to be different than that used with Fibonacci due to ERMT incapability of providing any time of service guarantee. And that is the *third* modification to the approach. Another important *fourth* modification, since ERMT is a *pull* technique (versus the *push* Fibonacci), streams can not start without serving any client, which means a number of clients equal to the number of the video streams must be served and are exempted from the defection prediction process. Assuming $\lambda_i > \lambda_{S_i}$, which should be the reason we reached this step in the process the modified more accurate predictions for DP_i and $ACWT_i$ are:

$$DP_i = \frac{\lambda_i - \lambda_{S_i}}{\lambda_i} \int_0^{l_i} F(x).dx, \tag{11}$$

and

$$ACWT_i = \frac{\lambda_i - \lambda_{S_i}}{\lambda_i} \frac{l_i}{2}. \tag{12}$$

The accuracy of ERMT defection rate prediction is presented in Figure 2(c). The prediction is not as accurate as with Fibonacci due to ERMT dynamic behavior and the complexities explained above.



(a) ERMT: Average Server Channel (TVOD)

(b) Fibonacci: Defection Probability

(c) ERMT: Defection Probability (120 Requests/Minute, FCFS)

Figure 2: Metric Prediction Accuracy [120 Videos, Video Length = 120 Minutes]

0.3.3 Step III: Distributing Fibonacci Resources

One trivial solution is to distribute the channels evenly among the group videos. Another better solution is to distribute the channels based on video request rates. That is similar to the approach with ERMT in predicting the video channel share, except the distribution equation is:

$$Ch_i = \frac{\lambda_i \times D_i}{\sum_{v=V_{ERMT}}^{V} (\lambda_v \times D_v)} \times Ch_{FB}. \tag{13}$$

In case of Fibonacci, the number of channels is independent from the request rate, but the defection rate of the video is a function of the available channels and the video length, and the video impact on the overall defection is directly related to its request rate compared to the overall group request rate. Nevertheless, since with Fibonacci we have pretty accurate prediction of the defection probability and average customer waiting time, an even better approach is to distribute the channels based on these predictions. That is to allocate a channel by channel to the video that gains the most from that channel in reducing its weighted defection rate, weighted by its request rate. And when/if all videos are predicted to have 0-D, the allocation should be based on the average customer waiting time weighted gain. The process stops only by allocating all group channels among group videos.

Table 2 summarize the symbols used in the algorithm description, while the algorithm is summarized in Table 3.

0.4 Statistical Cache Management (SCM)

The goal of cache management is to reduce the load on the storage subsystem and thus reduce the cost of the overall solution by caching selected data in the main memory. While *Interval Caching* [8] shown to be effective in reducing the I/O requirements when applied with Batching, it does not address by itself the problem of the limited network bandwidth and its effectiveness was not evaluated with more aggressive resource sharing techniques. Besides it can incur significant overhead on the system.

We investigate here a statistical approach for cache management. With this approach, the server periodically computes video access frequencies and determines the data to be cached based on these statistics. Besides being performance effective, this approach is easy to implement and incurs small overhead as updates are triggered only when the workload varies considerably.

In this section, we first derive the access distribution equations for videos served by various techniques and then use these equations to determine the optimal cache allocation. For ease of reference, Table 4 describes the main parameters used in the subsequent analysis.

Table 4: Main Parameters

Parameter Name	**Symbol**
Request Arrival Rate for the j^{th} Video (Request/s)	λ_j
Inter Arrival Time for the j^{th} Video (s) = $1/\lambda_j$	Δ_j
Frequency the t^{th} point of time from the j^{th} Video is requested (Request/s)	$F_j(t)$
Cache Size (s)	S
Last point of time from the j^{th} Video been Cached = Size of Cached Portion from the j^{th} Video (s)	S_j
Number of Videos	V
j^{th} Video Duration (s)	D_j
Regular Window for the j^{th} Video (s) (Patching and Transition patching)	Wr_j
Transition Window for the j^{th} Video (s) (Transition patching)	Wt_j
Number of Requests per Video Length for the j^{th} Video = $\lambda_j \times D_j$	N_j

0.4.1 Video Access Distributions

We derive next the access distribution equations for each playback point in a video delivered using various techniques. We assume here a TVOD model where every request is served immediately (except for Periodic Broadcasting). Hence, the average service rate of video j is the same as its average request arrival rate (λ_j). We also assume that $F_j(t)$ is the rate (frequency) by which the t^{th} point of time of video j is requested from the disks when there is no cache. The derived equations were validated by simulation. In Figures 3(a) through 3(d) we report the results for a video with a request arrival rate of 2 requests per minute (which is the rate for the second most popular video in the mixed workload, discussed in Section 0.6).

Patching and Controlled Multicast

Because of having two types of streams in Patching (and Controlled Multicast), and having a limit on the maximum patch length (Wr_j), the video can be divided into two regions: $[0, Wr_j]$ and $[Wr_j, D_j]$. Since every regular stream delivers the full video data, all points in the second region will be requested with an equal rate: $1/Wr_j$. The situation is different for the first region because the patches vary in length. The closer the point to the beginning of the video, the more likely it will be missed and requested by patches. The access rate approaches the video request rate (λ_j) as the point becomes closer to the beginning. Consequently, the access distribution can be formulated as follows:

$$F_j(t) = \begin{cases} \lambda_j - (\frac{\lambda_j}{Wr_j} - \frac{1}{(Wr_j)^2})t & 0 \leq t < Wr_j, \\ \frac{1}{Wr_j} & Wr_j \leq t \leq D_j. \end{cases} \tag{14}$$

Figure 3(a) shows that the analytical and simulation results match almost perfectly.

Transition Patching

Transition Patching has three types of streams: patches with maximum length of Wt_j, transition patches with lengths between $3Wt_j$ and $Wr_j + 2Wt_j$, and the regular streams with full video length. Therefore, the region $[Wr_j + 2Wt_j, D_j]$ is delivered by only regular streams at an access rate of $1/Wr_j$. As in Patching, the rate for region $[0, Wt_j]$ decreases linearly from λ_j

to $1/Wt_j$. The region $[Wt_j, 3Wt_j]$ is delivered by every transition patch and regular stream, and thus the rate is $1/Wt_j$. The region $[3Wt_j, 4Wt_j]$ is delivered by every transition patch and regular stream except the first transition patch in every regular window (Wr_j). The region $[4Wt_j, 5Wt_j]$ is delivered by every transition patch and regular stream except the first two transition patches in every regular window. The pattern continues up to the region $[Wr_j + Wt_j, Wr_j + 2Wt_j]$, which is delivered by only regular streams and the last transition patch in every regular window, leading to a stair shape curve. Consequently, the access distribution can be formulated as follows:

$$F_j(t) = \begin{cases} \lambda_j - (\frac{\lambda_j}{Wt_j} - \frac{1}{(Wt_j)^2})t & 0 \le t < Wt_j, \\ \frac{1}{Wt_j} & Wt_j \le t < 3Wt_j, \\ \frac{1}{Wt_j} - \frac{n}{Wr_j} & 3Wt_j \le t < Wr_j + 2Wt_j, \\ \frac{1}{Wr_j} & Wr_j + 2Wt_j \le t \le D_j, \end{cases} \tag{15}$$

where $n = \lfloor \frac{t}{Wt_j} \rfloor - 2$. Equation (15) was validated by simulation and the error is very small, as shown in Figure 3(b).

ERMT

Let us reexamine Figure 1(c) and assume that the inter-arrival time between successive requests for video j is Δ_j. (In reality, a customer does not arrive every exactly Δ_j, but this will be the average behavior over a long period.) Thus, 4 out of the 8 streams ($fraction = \frac{1}{2}$) have length $1 \times \Delta_j$, 2 streams ($fraction = \frac{1}{4}$) have length $4 \times \Delta_j$, and 1 stream ($fraction = \frac{1}{8}$) have length $10 \times \Delta_j$ on the average. This pattern can be generalized as follows: for N_j requests per video length, $\frac{N_j}{16}$ streams of the N_j streams are with length $22 \times \Delta_j$, $\frac{N_j}{32}$ with length $46 \times \Delta_j$, $\frac{N_j}{2^i}$ with length $L(i) \times \Delta_j$, and so on till $i = \log_2(N_j)$, where $L(i) = 2 \times L(i-1) + 2$, and $L(1) = \Delta_j$. Hence, the first part of the video from the beginning to Δ_j is delivered (i.e. requested) by every stream, with a request frequency equal to λ_j. The second part from Δ_j to $4 \times \Delta_j$ is requested by $N_j - N_j/2$ streams. The third part from $4 \times \Delta_j$ to $10 \times \Delta_j$ is requested by $N_j - N_j/2 - N_j/4$ streams, and so on. To generalize, the access frequency for the t^{th} point of time in the j^{th} video is given by

$$F_j(t) = \frac{\lambda_j}{2^{(1+\lfloor \log_2 (t/\Delta_j + 2)/3 \rfloor)}}. \tag{16}$$

Figure 3(c) shows that the analytical and simulations results do not match as closely as with the previous techniques because of the complex behavior of ERMT. Fortunately, for cache allocation only the relative access distribution for a video (compared with the others) does matter. To verify, we produced the results for ERMT in Section 0.6 with both the analytical model and the experimental results of access distribution collected during the simulations. The results were almost identical with an error well below 1%.

Periodic Broadcasting

The access distribution function $F_j(t)$ is very simple in the case of periodic broadcasting. For each video segment, it is the inverse of the segment repeat time. The segment repeat time is simply the segment length with GDB, SB, and FB and the segment length divided by the segment allocated bandwidth with HB.

Catching and Selective Catching

In Catching, each video is divided into K_j segments, and each segment is broadcast periodically on a dedicated channel. Thus, the access frequency for any point in a segment is equal to the inverse of its length. Catching also uses patch streams to service requests immediately. Patches produce additional accesses to the data points of the first segment. The request frequency due to patches decreases linearly up to the last point in the first segment. Consequently, the access distribution is given by

$$F_j(t) = \begin{cases} \lambda_j + \frac{h(K_j)}{D_j} - \lambda_j \frac{h(K_j)}{D_j} t & 0 \leq t \leq \frac{D_j}{h(K_j)}, \\ \frac{h(K_j)}{f(n) D_j} & \frac{D_j}{h(K_j)} < t \leq D_j, \end{cases} \tag{17}$$

where $h(m) = \Sigma_{i=1}^{m} f(i)$, $f(i)$ is the i_{th}-segment's relative size compared to the first segment in the modified GDB partition series, and n can be found using the inverse of function $h(m)$ as $n = h^{-1}(\frac{h(K_j)}{D_j} t)$. Figure 3(d) shows that the analytical and simulation results match almost perfectly.

For Selective Catching, Equation (17) can be used for hot videos while Equation (14) can be used for cold videos.

WAHS

For WAHS, Equation (16) can be used for the ERMT videos, while the calculations discussed in Subsection (0.4.1) applies for Fibonacci videos.

0.4.2 Cache Allocation Model

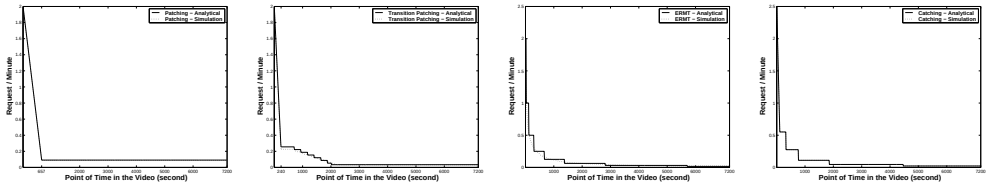
The derived access distribution models in the previous subsection exhibit the following property: $F_j(t_1) \geq F_j(t_2)$ for all $t_1 < t_2$. If the optimal cached size for the j^{th} video is found to be S_j seconds, then we simply cache from the beginning of the video till the $S_j{}^{th}$ second. To allocate the available cache optimally among all videos under such property, we simply need to solve two equations:

$$S = \sum_{j=1}^{V} S_j, \tag{18}$$

and

$$F_i(S_i) = F_j(S_j), \tag{19}$$

for all $i, j \in VideosSet$ with S_i and $S_j > 0$. Equation (19) means that for all videos with cached portions greater than zero, the request frequency of the last cached frame for every video must be equal (or very close).



(a) Patching (b) Transition Patching (c) ERMT (d) Catching

Figure 3: Access Distribution $F_j(t)$

0.5 Tunings of Design Parameters

In this section, we derive equations for determining optimally the values of W_r and W_t for Transition Patching. For Selective Catching, we present an approach for deciding on what videos to serve with Catching and what

videos with Controlled Multicast when the server resources are limited. No approach was provided in [16] for such situations. For the other techniques, we highlight important results reported in [19, 16, 15].

0.5.1 Transition Patching

The objective here is to derive simple equations for Wr and Wt, which minimize the number of required server channels. The number of required channels (ch_j) with any technique is given by

$$ch_j = \int_0^{D_j} F_j(t)dt, \tag{20}$$

where $F_j(t)$ is the request distribution for the chosen technique. For Transition Patching,

$$ch_j = \frac{1}{2}\lambda_j Wt_j + \frac{1}{2}\frac{Wr_j}{Wt_j} + \frac{D_j}{Wr_j} - 2\frac{Wt_j}{Wr_j} + 1. \tag{21}$$

By taking the first derivative of Equation (21) with respect to Wr_j and equating it with zero, we get the optimal value of Wr_j:

$$Wr_j = \sqrt{2Wt_j(D_j - 2Wt_j)}. \tag{22}$$

Since the transition patch cannot exceed the video length (i.e., $Wr_j + 2Wt_j < D_j$) and $Wt_j < Wr_j$, it follows that $Wt_j < \frac{1}{3}D_j$.

Similarly, to find the optimal value of Wt_j, we take the first derivative of Equation (21) with respect to Wt_j and equate it with zero, and then substitute the value of Wr_j from Equation (22):

$$Wt_j = \sqrt[3]{2D_j\Delta_j^2} / \sqrt[3]{1 - 2Wt_j/D_j}. \tag{23}$$

Typically, $D_j >> 2Wt_j$, and thus

$$Wt_j \approx \sqrt[3]{2D_j/\lambda_j^2} = \sqrt[3]{2\Delta_j^2 D_j}. \tag{24}$$

Alternatively, we can substitute this value in Equation (23) and get the following equation that is more accurate:

$$Wt_j \approx \sqrt[3]{2\Delta_j^2 D_j} / \sqrt[3]{1 - 2\sqrt[3]{2\Delta_j^2/D_j^2}}. \tag{25}$$

To avoid a negative value of Wt_j, $N_j = \lambda_j D_j > 4$. (It can be shown that for Transition Patching to be more cost effective than Patching, $N_j = \lambda_j D_j \geq 12$.) By optimizing the number of channels Equation (21) with respect to Wr_j and Wt_j, and assuming $Wt_j << D_j$, we obtain:

$$Wr_{j_{opt}} = \sqrt{2Wt_j(D_j - 2Wt_j)}$$

and

$$Wt_{j_{opt}} \approx \sqrt[3]{2\Delta_j^2 D_j} / \sqrt[3]{1 - 2\sqrt[3]{2\Delta_j^2/D_j^2}}.$$

Note that for Transition Patching to be more cost effective than Patching, $N_j = \lambda_j D_j \geq 12$.

0.5.2 Patching and Controlled Multicast

In [19], an optimal value for Wr_j in Patching/Controlled Multicast was derived:

$$Wr_j = \sqrt{2D_j/\lambda_j} = \sqrt{2\Delta_j D_j}. \tag{26}$$

Based on that, the number of required channels can be given by

$$ch_j = \sqrt{2\lambda_j D_j} - 1/2. \tag{27}$$

A slightly different equation was reached in [15], but our analysis favors Equation (26). Equations (27) and (26) assume $N_j = \lambda_j D_j > 1$; otherwise, no two streams will overlap and a unicast is the only option.

0.5.3 Catching and Selective Catching

In [15] and [16], the optimal number of server channels required for Controlled Multicast/Patching and for Catching were shown to be

$$ch_j = \sqrt{2\lambda_j D_j + 1} - 1 \tag{28}$$

and

$$ch_j = K_j^* + \lambda_j FS_j/2, \tag{29}$$

respectively. Where FS_j is the first segment length of video j and K_j^* is the optimal number of broadcast channels. This equation can be optimized with respect to K_j to find the optimal value K_j^*. In Selective Catching, a key issue

is how to decide whether a video is hot or cold. One possible approach is to compare the numbers of required channels to serve the video with Catching and with Controlled Multicast [16]. The video is considered hot if the first number is smaller and cold otherwise. By comparing the required channels by both, the technique that results in smaller value is chosen for service [16]. Unfortunately, in many situations the number of server channels is not sufficient to satisfy the Catching requirement for every hot video. In that case we need to determine the subset of the hot videos to be served by Catching. We introduce the following *additional* test: video j is served by Catching if

$$ServerChannels \times \frac{\lambda_j}{\lambda} \geq ch_j. \tag{30}$$

The idea here is to serve by Catching, among the hot videos, only the videos whose numbers of required channels are not greater than their fair share, determined by their contributions to the total number of requests.

0.6 Performance Evaluation

0.6.1 Evaluation Methodology

We have developed a simulator for VOD that models various resource sharing techniques and captures the derived analytical models for cache management. The simulator has been validated by reproducing graphs in previous studies. Simulation stop after a steady state analysis with 95% confidence interval is reached.

We consider two service models: *True Video-on-Demand* (TVOD) and *Near Video-on-Demand* (NVOD). We also consider two video workloads: *mixed-video* and *hot-video*. The mixed-video workload contains videos with popularities varying from cold to hot, and the hot-video workload contains only hot videos. The later is used mainly to evaluate periodic broadcasting techniques which are designed mainly for popular videos. The two service models and the two video workloads lead to four target environments: *mixed-video workload and TVOD*, *hot-video workload and TVOD*, *mixed-video workload and NVOD*, and *hot-video workload and NVOD*.

In the TVOD model, we compare the server resources required to service all requests immediately. In the NVOD model, however, we fix the server resources and compare the *customer defection probability* (defined as

the probability that customers leave the server without being serviced because of waiting times exceeding their tolerance), the *average waiting time*, and *unfairness* towards cold videos. The first metric translates directly to the number of customers that can be serviced concurrently and to server throughput. Two types of server resources are considered: *server channels* and *disk channels*. Server channels represent the requirement in network bandwidth, whereas disk channels represent the disk I/O bandwidth requirement. A channel is capable of supporting only one stream at a time. Without cache in the main memory to reduce the load on the storage subsystem, the number of disk channels is equal to the number of server channels.

0.6.2 Workload Characteristics

In accordance with most prior studies, we assume that the arrival of requests follows a Poisson Process with an average arrival rate λ and that the accesses to videos are highly localized and follow a Zipf-like distribution with skewness parameter $\theta = 0.271$. The mixed-video workload contains 120 2-hour videos, whereas in the hot-video workload we consider three numbers of videos: 1, 10, and 20 2-hour each. Table 5 shows the values of the input parameters in both workloads.

We consider two models of customer waiting tolerance. In general, we assume as in prior work that the tolerance follows the exponential distribution with a mean value of 2 minutes. While with periodic broadcasting techniques, we borrow and apply the *Time of Service Guarantee* (TSG) model [31, 34], which was used in a different context, since broadcasting techniques can provide time of service guarantees by upper limiting the waiting times by the length of the first segment. With TSG, customers who receive time of service guarantees shorter than 2 minutes will wait to be serviced, whereas the waiting tolerance of all other customers follows the exponential distribution with a mean value of 2 minutes.

For scheduling, we use *Maximum Queue Length* (MQL) [10] and *First Come First Serve* (FCFS) [10]. The results for *Maximum Factored Queue Length* (MFQL) [1] are not shown because it performs poorly in the streaming merging environment (despite its relatively good performance with Batching).

Table 5: Parameter Values

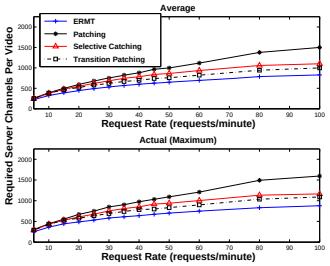
Parameter	**Mixed-Video**	**Hot-Video**
Request Arrival Rate (Req./min)	5 - 240	60 - 2400
Number of Videos	120	$1, 10, 20$
Video Duration (min)	120	
Video Skewness (θ)	0.271	
Cache Space	5% of total size of videos	
Scheduling Policy	MQL	MQL, FCFS
Waiting Tolerance Model	Exponential, TSG	

0.6.3 Traditional Techniques

Environment I: Mixed-Video and TVOD

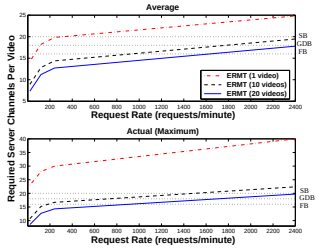
This environment helps in comparing stream merging and composite techniques in terms of the number of server and disk channels required to achieve TVOD (i.e. zero defection and waiting time). In this environment, we vary the request arrival rate from 5 to 100 requests per minute, and we collect the average and the maximum server channels required when using the different techniques. The average results represent the average requirements over long periods, while the maximum results represent the actual required number of channels for providing TVOD for all requests. Figure 4(a) plots these results. As expected, the gaps among the techniques widen as the arrival rate increases because the workload offers increasingly higher degrees of resource sharing, which are exploited to different levels by different techniques. Generally, ERMT has the lowest requirements, and Patching the highest. Interestingly, Selective Catching does not perform relatively well.

We experimented with a modified version of Selective Catching that employs ERMT rather than Patching for stream merging, but the performance benefits are low and may not justify the increased complexity. These results are due to the outstanding merging abilities of ERMT, especially for hot videos. The figure also illustrate that resource sharing can achieve significant reductions in network bandwidth requirements. Compared with the simple unicast scheme, the reductions in the average network bandwidth with ERMT approximately range from a factor of 3 to a factor of 15 as the arrival rate increases.

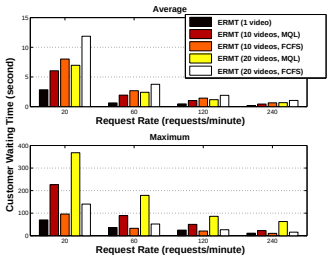


(a) Effect of Request Rate on Required Channels

Figure 4: Environment I



(a) Effect of Request Rate on Required Channels



(b) Effect of Request Rate on Waiting Time (Average Required Channels Provided)

Figure 5: Environment II

Environment II: Hot-Video and TVOD

In this environment, we compare the performance of various periodic broadcasting techniques (SB, GDB, and FB) and the best performer in Environment I (ERMT). Since periodic broadcasting cannot provide a zero waiting time, we assume that a maximum waiting time below 2 seconds constitutes a TVOD service.

Figure 5(a) plots the average and maximum required server channels per video to provide TVOD for the different techniques. Note that unlike periodic broadcasting, the number of server channels per video with ERMT depends on the arrival rate and the number of supported videos. Whereas periodic broadcasting techniques reserve channels for each video, ERMT deals with all the resources as one pool and distributes them dynamically to the requests. While the average helps in understanding the overall system load, the maximum indicates the actual bandwidth to be reserved to achieve TVOD. The distinction between the two metrics was always ignored in previous studies, where only the average results were usually reported. The results demonstrate that ERMT scales very well with the increase in request rate even in terms of both the average and actual (maximum) required bandwidth. For a workload of 20 2-hour videos, the request rate must be constantly over 900 requests/minute for the best periodic broadcasting technique (FB) to be a better choice than ERMT. Even when that is the case, while ERMT requires more guaranteed bandwidth, it does not consume it all the time, which may enable the system to face short periods of drops in server capacity. Since the required bandwidth with ERMT is a function of the number

of requests per video length (i.e., $N = \lambda D$), the breakpoint is higher than 900 requests/minute when the videos are shorter.

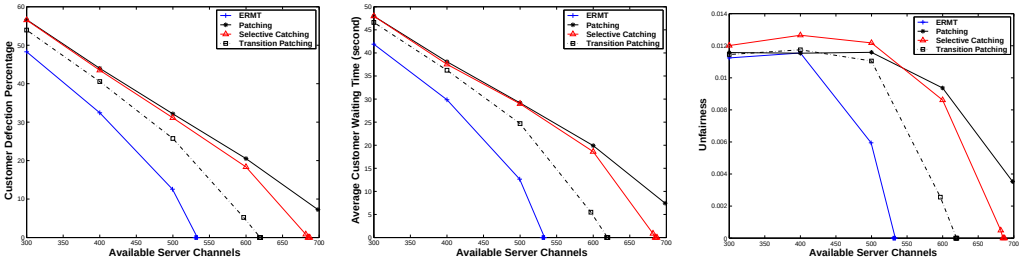
To highlight the effect of the difference between the average and actual required number of channels, Figure 5(b) plots the average and maximum customer waiting times if only the average required bandwidth is provided. Therefore, the number of provided channels varies with the request rate and with the number of videos as well. The service is obviously not TVOD, especially for the relatively low request rates since some customers had to wait few minutes for service. Since scheduling policies can impact the performance of ERMT compared with periodic broadcasting, both MQL and FCFS are examined. MQL achieves higher throughput and shorter average waiting time, whereas FCFS is fairer and can significantly reduce the maximum waiting time.

Environment III: Mixed-Video and NVOD

By varying the number of server channels, we can use this environment to compare stream merging and composite techniques in terms of the achieved average customer defection probability, waiting times, and unfairness. Figure 6 plots these three metrics versus the number of server channels for various techniques. The results here demonstrate once again that ERMT performs the best and Patching the worst among stream merging and composite techniques. There is a clear correlation among the three metrics. The correlation between unfairness and the other two metrics is due to using Maximum Queue Length (MQL) [10] for scheduling. MQL selects for service the longest video queue, trading fairness for performance. The more limited the number of channels becomes, the longer customers have to wait, and the faster the waiting queues build up. Consequently, both the defection probability and the bias against cold videos increase.

Environment IV: Hot-Video and NVOD

In this environment, we also compare the performance of various periodic broadcasting techniques with ERMT. We consider here two waiting tolerance models: exponential and TSG (discussed in Subsection 0.6.2). The later captures the capability of periodic broadcasting techniques to limit the maximum waiting time and predict the actual times of service. By informing clients upon arrival with their times of service, the server can encourage them



(a) Effect of Server Channels on Defection Probability (b) Effect of Server Channels on Average Waiting Time (c) Effect of Server Channels on Unfairness

Figure 6: Environment III [30 Requests/Minute]

to wait.

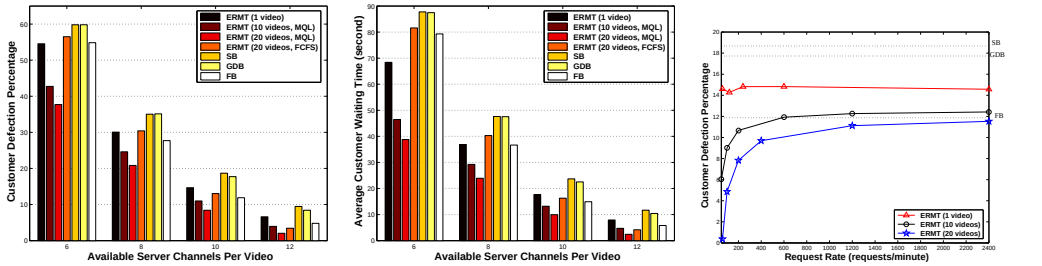
Figures 7(a) through 7(d) depict the defection probability and average waiting time under the exponential waiting tolerance model for different server capacities, request rates, and numbers of videos. These results confirm that ERMT is very competitive, compared with the best periodic broadcasting technique, even in servicing very hot videos and under very limited resources.

Figures 7(e) and 7(f) compare various techniques in terms of the defection probability and average waiting time under the TSG waiting tolerance model, which captures the ability of the broadcasting techniques in providing waiting time guarantees. The results demonstrate that broadcasting techniques can achieve much lower defection probabilities than ERMT, especially when the available server bandwidth is not very limited. With only 8 server channels per video, all broadcasting techniques cause no defections, whereas ERMT causes more than 20% defections. However, ERMT achieves shorter average waiting time for those clients who were serviced, partially due to the high defection percentage.

Cautious Harmonic Broadcasting (CHB)

Despite its high effectiveness in reducing server bandwidth, Cautious Harmonic Broadcasting (CHB) requires high client bandwidth. The required number of client channels is equal to the number server channels allocated for the video. In contrast, FB, SB, GDB, and ERMT always require only two channels. Moreover, CHB partitions each supported video into a large number of segments. For example, to achieve a maximum waiting time of

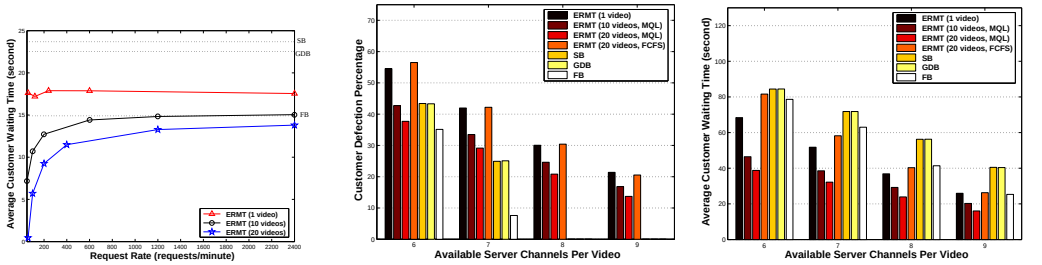
5 seconds in serving a 2-hour video, CHB divides the video into 1440 segments broadcast on 1339 channels with aggregate bandwidth of 8.34 times the video playback rate, while FB, SB, and GDB divide the video into only 14, 17, and 16 segments, respectively. More importantly, CHB (or QHB) requires each client to have download bandwidth higher than 8 times the video playback rate, while the other three broadcasting techniques (and ERMT as well) require client bandwidth of only double the video playback rate.



(a) Effect of Server Channels per Video on Defection Probability (Exponential)

(b) Effect of Server Channels per Video on Average Waiting Time (Exponential)

(c) Effect of Request Rate on Defection Probability (Exponential)



(d) Effect of Request Rate on Average Waiting Time (Exponential)

(e) Effect of Server Channels per Video on Defection Probability (TSG)

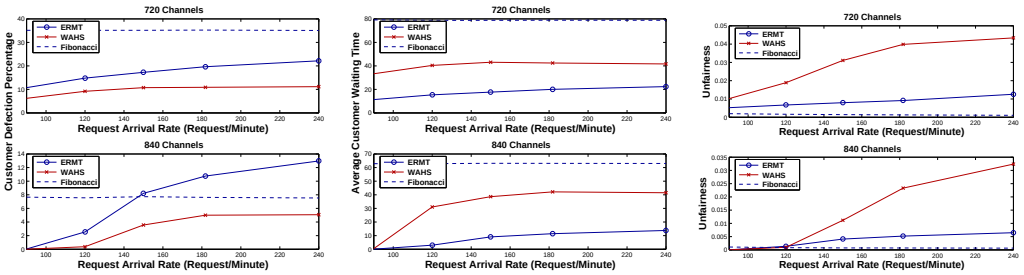
(f) Effect of Server Channels per Video on Average Waiting Time (TSG)

Figure 7: Environment IV [Default: 240 Requests/Minute, 10 Channels per Video, MQL]

0.6.4 WAHS: The Proposed Hybrid Solution

Now, we are ready to evaluate WAHS compared to the individual techniques forming it, which are, as shown earlier, the best each in its class. The realistic mixed workload is used in this comparison.

Figure 8 demonstrates a significant gain in the server throughput by using WAHS over using ERMT or Fibonacci individually. WAHS can eliminate more than half the defections compared to the best alternative among ERMT and Fibonacci, and more than 70% compared to the other. ERMT keeps its lead in terms of customer waiting time calculated for customers who actually received service, but that is on the expense of serving less customers. In addition, with WAHS, the majority of customers do receive some sort of time of service guarantee, as shown in Figure 9, which can improve the perceived quality of service by customers even when they do wait longer, while with ERMT no customer receives any time of service guarantee. Fibonacci, as expected, remains the best in fairness due to its blindness to video popularity. Finally, it worths mentioning that under very high request rates with sufficient server capacity WAHS converges to Fibonacci periodic broadcasting, while under very light workloads it converges to ERMT.



(a) Defection (b) Waiting (c) Unfairness

Figure 8: WAHS Compared to ERMT and Fibonacci [MQL, TSG]

0.6.5 SCM: The Proposed Cache Management

Finally, let us evaluate the effectiveness of the SCM and compare it to Interval Caching. Figures 10 and 11 present the reduction in I/O requirements (average and maximum) due to server side cache for both SCM and Interval caching when used with the best three resource sharing techniques: WAHS, ERMT, and Fibonacci.

SCM is very effictive in reducing the average I/O requirement with all three resource sharing techniques, and the maximum I/O requirement with WAHS and Fibonacci. the reduction in all five cases is between 10% and 18% with as little cache space as 2% of the total video size. However, it is

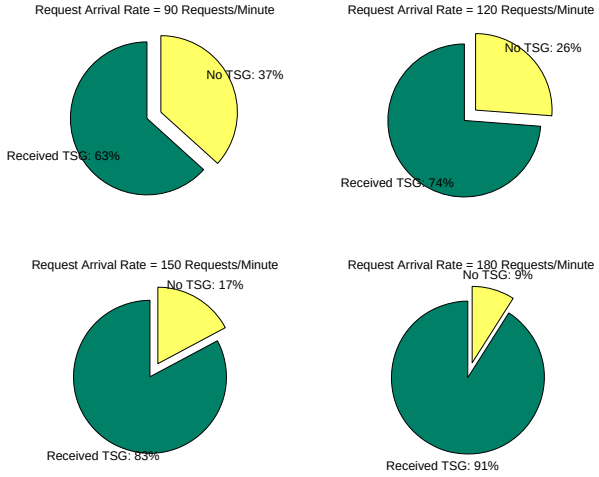
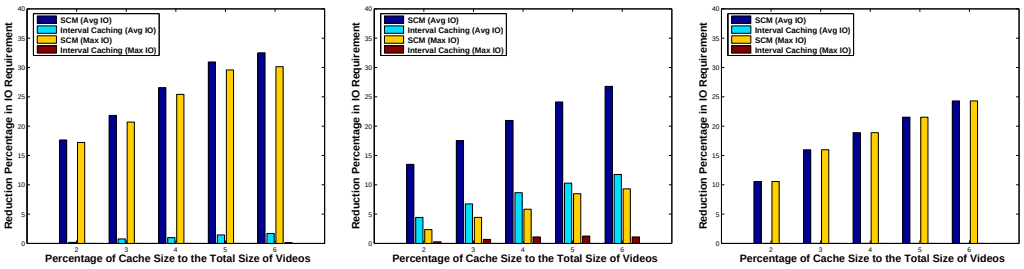


Figure 9: Ratio of Customers Received Time of Service Guarantee [WAHS, 720 Channels, MQL]

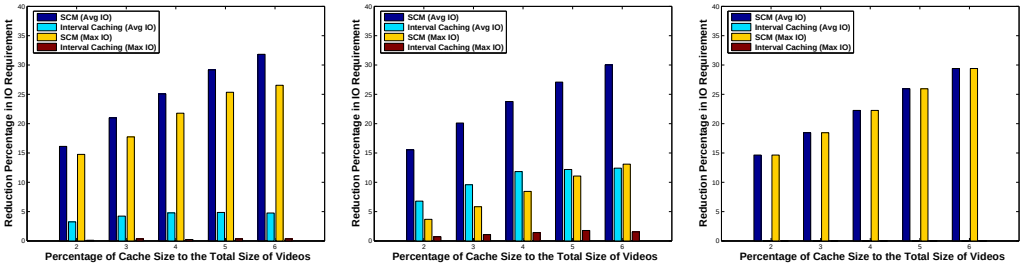
not as efficient in reducing the maximum I/O requirement with ERMT. That is due to ERMT dynamic nature and irregualrity in resource consumption and allocation among videos. But even in that worst case, and of course, in the other better five cases, it remains significantly better that the Interval Caching alternative. For example, when SCM is used with WAHS with a cache size 6% of the total video size and a server capacity equal to 720 channels, the reduction easily exceeds 30%, while Interval Caching fails to hit the 2% reduction mark.

SCM was evaluated with the other less effecient techniques in both classes and found to be effictive with them, as well. Figure 12 plot some results for the stream merging group and selective catching. ERMT requires the lowest average disk I/O channels, and Patching the highest. However, as expected, the effectiveness of caching decreases, and the gaps among various techniques diminish as we keep increasing the cache size. An interesting observation in Figure 12(c), which depicts the actual numbers of disk channels that must be available to provide TVOD for all requests. Whereas ERMT requires the smallest number of disk channels with no cache, the situation changes when the cache is introduced. With a cache size greater than 5% of the total size of videos, Transition Patching starts requiring fewer disk channels than ERMT. With a cache size greater than 12%, all the other techniques require fewer disk channels than ERMT. ERMT benefits the least from cache



(a) WAHS (b) ERMT (c) Fibonacci

Figure 10: Impact of Cache Model and Size on IO Requirement, [720 Channels, 150 Requests/Minute, MQL, TSG]



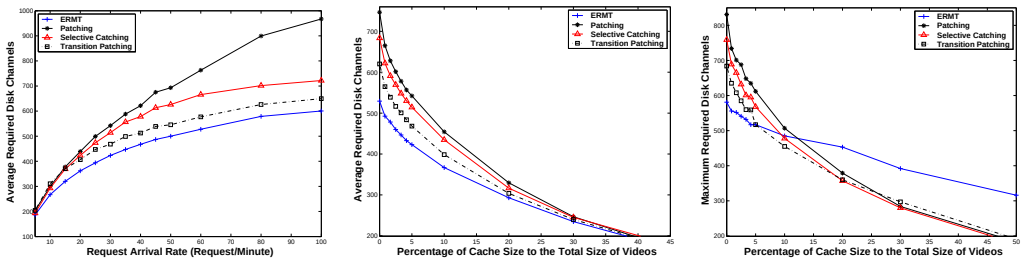
(a) WAHS (b) ERMT (c) Fibonacci

Figure 11: Impact of Cache Model and Size on IO Requirement, [840 Channels, 150 Requests/Minute, MQL, TSG]

and Patching benefits the most because of the access distribution curves in Subsection 0.4.1. In particular, as the cache size increases from 1% to 2%, ERMT reduces the average disk channel requirement by approximately 7% to 10% and the actual disk channels required to provide TVOD by 4% to 5%, compared with 11% to 16% and 12% to 16%, respectively, with Patching.

0.7 Conclusions

This study has provided four main contributions. First, we have conducted a detailed analysis of major resource sharing techniques, spanning different classes, and have developed analytical models for optimal tuning of design parameters for some techniques. Second, we have proposed a Workload Aware Hybrid Solution (WAHS) that combines the advantages of stream merging



(a) Effect of Request Rate on Average Disk Channels (b) Effect of Cache Size on Average Disk Channels (c) Effect of Cache Size on Maximum Disk Channels

Figure 12: Impact of Cache on IO Requirement [Environment I, 30 Requests/Minute]

with periodic broadcasting and adapts dynamically to changes in workload and server resources. Third, we have proposed a statistical cache management approach (SCM) and have derived analytical models for optimal cache allocation to reduce the demands on the disk I/O when various resource sharing techniques are used. Fourth, we have introduced the load on the underlying storage subsystem as an additional dimension to the comparisons among various techniques and have examined the impact of caching on reducing these requirements.

In the evaluation, we have divided various techniques into two sets. The first set contains techniques that are suitable for environments with both hot and cold videos: Patching, Transition Patching, ERMT, and Selective Catching. The second set contains techniques that are suitable for environments with only hot videos. These techniques include the best performer of the first set (which turned out to be ERMT) and three periodic broadcasting techniques: GDB, SB, and FB. Then we compared WAHS with the best technique from each group. We have considered two service models: *True Video-on-Demand* (TVOD) and *Near Video-on-Demand* (NVOD).

The main results can be summarized as follows.

- Resource sharing can achieve significant reductions in both the required network bandwidth and the disk I/O bandwidth, compared with the simple unicast scheme. These reductions can be up to a factor of 60 for very high request arrival rates.

- In workloads containing both hot and cold videos, and among traditional techniques, ERMT performs the best in general. It requires the

smallest number of server channels for providing a TVOD service. It also achieves the highest server throughput (or the least defection probability) and the shortest average customer waiting time in the NVOD service model. However, Transition Patching can achive better utilization of relatively large cache sizes and as a result, it requires lower disk I/O bandwidth than ERMT under such conditions.

- In workloads containing only hot videos, and among existing techniques, FB performs better than both SB and GDB in terms of the waiting times. HB is the most effective in reducing the server bandwidth, but it requires clients to have much more download bandwidth and to listen simultaneously to a much larger number of multicast groups. ERMT is practically better capable of providing TVOD than periodic broadcasting. By assuming that a waiting time smaller than 2 seconds constitutes TVOD in the case of periodic broadcasting, FB requires less server bandwidth per video than ERMT only for excessively high request arrival rates or when the number of supported videos is rather small. For example, with 20 2-hour videos, FB requires less cost than ERMT only if the request arrival rate is higher than 900 requests per minute. That arrival rate must be even higher when the videos are shorter in length. To achieve NVOD, FB achieves higher throughput than ERMT when customers are encouraged to wait, by informing them with the expected times of service. ERMT, however, performs better in terms of the average waiting time.

- WAHS out perform both ERMT and FB in server throughput. In fact, WAHS can eliminate more than half the defections compared to the best alternative among ERMT and Fibonacci, and more than 70% compared to the other. WAHS out perform, as well, FB in average customer waiting time before service. While ERMT can result in shorter waiting times, but unlike WAHS it fails to provide any sort of time of service guarantee. With WAHS, the majority of customers do receive some sort of time of service guarantee.

- SCM is very effective in reducing the disk I/O bandwidth requirements further: with a cache size of only 2% of the total size of videos, these requirements can be reduced by up to 18%. And with a cache size of 6%, the reduction can exceed 30%.

Consequently, WAHS is the best choice under wide range of workloads and server resource levels. However, WAHS can dynamically converge to either ERMT or FB only when that is expected to result in a better performance. SCM is highly recommended with WAHS to reduce further the load on the underlying disk I/O bandwidth.

Bibliography

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. The maximum factor queue length batching scheme for Video-on-Demand systems. *IEEE Trans. on Computers*, 50(2):97–110, Feb. 2001.

[2] O. Bagouet, K. A. Hua, and D. Oger. A periodic broadcast protocol for heterogeneous receivers. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, Jan. 2003.

[3] A. Bar-Noy, G. Goshi, R. Ladner, and K. Tam. Comparison of stream merging algorithms for Media-on-Demand. *Multimedia Systems Journal*, 9:211–223, 2004.

[4] M. Bradshaw, B. Wang, S. Sen, L. Gao, J. Kurose, P. Shenoy, and D. Towsley. Periodic broadcast and patching services: Implementation, measurement and analysis in an Internet streaming media testbed. In *Proc. of ACM Multimedia*, pages 280–290, Oct. 2001.

[5] Y. Cai and K. A. Hua. An efficient bandwidth-sharing technique for true video on demand systems. In *Proc. of ACM Multimedia*, pages 211–214, Oct. 1999.

[6] Y. Cai and K. A. Hua. Sharing multicast videos using patching streams. *Multimedia Tools and Applications journal*, 21(2):125–146, Nov. 2003.

[7] Y. Cai, W. Tavanapong, and K. A. Hua. Enhancing patching performance through double patching. In *Proc. of 9th Intl Conf. on Distributed Multimedia Systems*, pages 72–77, Sept. 2003.

[8] A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and caching in large-scale video servers. In *Digest of Papers. IEEE Int'l Computer Conf.*, pages 217–225, March 1995.

[9] A. Dan and D. Sitaram. A generalized interval caching policy for mixed interactive and long video workloads. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, pages 344–351, Jan. 1996.

[10] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. of ACM Multimedia*, pages 391–398, Oct. 1994.

[11] D. L. Eager, M. K. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for Video-on-Demand servers. In *Proc. of ACM Multimedia*, pages 199–202, Oct. 1999.

[12] D. L. Eager, M. K. Vernon, and J. Zahorjan. Bandwidth skimming: A technique for cost-effective Video-on-Demand. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, pages 206–215, Jan. 2000.

[13] D. L. Eager, M. K. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):742–757, Sept. 2001.

[14] L. Gao, J. Kurose, and D. Towsley. Efficient schemes for broadcasting popular videos. In *Proc. of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, July 1998.

[15] L. Gao and D. Towsley. Supplying instantaneous Video-on-Demand services using controlled multicast. In *Proc. of IEEE Multimedia Computing and Systems*, pages 117–121, June 1999.

[16] L. Gao, Z.-L. Zhang, and D. F. Towsley. Catching and selective catching: efficient latency reduction techniques for delivering continuous multimedia streams. In *Proc. of ACM Multimedia*, pages 203–206, Oct. 1999.

[17] L. Golubchik, J. C. S. Lui, and R. Muntz. Reducing I/O demand in Video-On-Demand storage servers. In *Proc. of ACM SIGMETRICS*, pages 25–36, May 1995.

[18] J. Gray and P. Shenoy. Rules of thumb in data engineering. In *Proc. of the IEEE International Conference on Data Engineering*, Feb 2000.

[19] C. Griwodz, M. Liepert, M. Zink, and R. Steinmetz. Tune to Lambda Patching. *ACM Performance Evaluation Review*, 27(4):20–26, March 2000.

[20] A. Hu. Video-on-Demand broadcasting protocols: A comprehensive study. In *Proc. of IEEE INFOCOM*, April 2001.

[21] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true Video-on-Demand services. In *Proc. of ACM Multimedia*, pages 191–200, 1998.

[22] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan Video-on-Demand system. In *Proc. of ACM SIGCOMM*, pages 89–100, Sept. 1997.

[23] C. Huang, R. Janakiraman, and L. Xu. Loss-resilient on-demand media streaming using priority encoding. In *Proc. of ACM Multimedia*, pages 152–159, Oct. 2004.

[24] L. Juhn and L. Tseng. Harmonic broadcasting for Video-on-Demand service. *IEEE Trans. on Broadcasting*, 43(3):268–271, Sept. 1997.

[25] J. Liu and J. Xu. Proxy caching for media streaming over the Internet. *IEEE Communications Magazine*, 42(8):88–94, Aug. 2004.

[26] J.-F. Pâris, S. W. Carter, and D. D. E. Long. Efficient broadcasting protocols for video on demand. In *Proc. of the Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 127–132, July 1998.

[27] B. Qudah and N. J. Sarhan. Towards scalable delivery of video streams to heterogeneous receivers. In *Proc. of ACM Multimedia*, pages 347–356, Oct. 2006.

[28] B. Qudah and N. J. Sarhan. Towards enhanced resource sharing in video streaming with generalized access patterns. In *ICME 2007*, July 2007.

[29] M. Rocha, M. Maia, I. Cunha, J. Almeida, and S. Campos. Scalable media streaming to interactive users. In *Proc. of ACM Multimedia*, pages 966–975, Nov. 2005.

[30] N. J. Sarhan and C. R. Das. Caching and scheduling in NAD-based multimedia servers. *IEEE Trans. on Parallel and Distributed Systems*, 15(10):921–933, Oct. 2004.

[31] N. J. Sarhan and C. R. Das. A new class of scheduling policies for providing time of service guarantees in Video-On-Demand servers. In *Proc. of the 7th IFIP/IEEE Int'l Conf. on Management of Multimedia Networks and Services*, pages 127–139, Oct. 2004.

[32] P. Sessini, L. Shi, A. Mahanti, Z. Li, and D. L. Eager. Scalable streaming for heterogeneous clients. In *Proc. of ACM Multimedia*, Oct. 2006.

[33] M. A. Tantaoui, K. A. Hua, and T. T. Do. Broadcatch: A periodic broadcast technique for heterogeneous Video-on-Demand. *IEEE Trans. on Broadcasting*, 50(3), Sept. 2004.

[34] A. Tsiolis and M. K. Vernon. Group-guaranteed channel capacity in multimedia storage servers. In *Proc. of ACM SIGMETRICS*, pages 285–297, 1997.