Study of the Impact of Replacement Granularity and Associated Strategies on Video Caching*

Elias Balafoutis, Antonis Panagakis, Nikolaos Laoutaris, Ioannis Stavrakakis
Department of Informatics & Telecommunications,
University of Athens, 15784 Athens, Greece
email:{balaf,apan,laoutaris,istavrak}@di.uoa.gr

Abstract

Partial caching of large media objects such as video files has been proposed recently as the caching of entire objects can easily exhaust the storage resources of a proxy server. In this paper the idea of segmenting video files into chunks and applying replacement decisions at the chunk level rather than on entire videos is examined. It is shown that a higher byte hit ratio (BHR) can be achieved by appropriately adjusting the replacement granularity. The price paid for the improved BHR performance is that the replacement algorithm takes a longer time to converge to the steady state BHR. For the segmentation of video into chunks two methods are presented. The Fixed Chunk Size segmentation scheme that is rather simple and reveals the basic trade-off between byte hit ratio (BHR) and responsiveness to changes of popularity; the Variable Chunk Size segmentation scheme that uses the request frequencies to dynamically adjust the size of the chunk and is shown to be capable of combining a small response time with high BHR. Moreover, a variation of the fixed chunk size segmentation scheme is presented, which is shown to improve its performance by switching between different chunk sizes. Video segmentation is also considered as a mechanism to provide for caching differentiation based on access costs. By employing access cost dependent chunk sizes an overall access cost reduction is demonstrated.

*Work supported in part by the General Secretariat for Research and Technology of Greece under the Joint Greek-Italian Scientific and Technological Cooperation Programme

1 Introduction

The explosive growth of demand for bandwidth, fuelled by the introduction of the world wide web, found data networks unprepared to handle the new traffic volumes. This led to an increase in both loss rates and user perceived latency, that could easily hamper the new global information delivery system. Caching, i.e, demand driven replication of data objects close to the clients, has been successfully used to relieve the backbone network and reduce the delivery delay of requested data. In a similar way contemporary networks, although copying adequately with web traffic, seem to have difficulty in managing effectively the delivery of information-rich content such as streaming video, which is rising as the new popular media to be integrated in the internet infrastructure. In the context of caching, the prominent characteristic of video, is it's large size. The large size of video makes the segmentation of video objects a sensible approach and has inspired the development of partial caching techniques which operate on segments of video files rather than on entire videos.

A variety of caching schemes have been proposed to handle video. Initial works on video caching have inherited the main characteristics of web caching and have treated videos as single entities which are either cached completely or not at all [1, 2, 3]. More recent works have considered the video characteristics such as the associated rate variability, structure and large size, and have investigated the idea of partial video caching. In [4] the initial frames of each video (called the prefix) are cached in the proxy in order to improve the startup latency experienced by users. Additionally, smoothing is performed to reduce the peak bandwidth and increase the utilization of leased network channels that connect the proxy with the origin server. In a similar approach in [5], the bursty part of a VBR video stream is selected to be stored at the proxy while the remaining smooth part is retrieved directly from the repository, thus reducing the peak bandwidth requirement in the backbone links. Both aforementioned schemes deal with the burstiness, which is inherent in VBR encoding algorithms.

In [6, 7] the prefix is stored in the cache and the remaining part (the suffix) is either explicitly requested from the video repository or retrieved through an ongoing multicast transmission which services a group of concurrent users; in the later case, a patch might be requested directly from the

repository, so as to fill the temporal gap between the cached prefix and the currently multicasted part of the suffix. The aim of these works is to reduce the volume of traffic that crosses the backbone; this is achieved by merging requests into multicast groups. Window based request merging is poroposed in [8, 9]. In particular, local proxies cache a sliding window of data, trying to merge requests for the same stream that arrive closely in time so that they may serviced from a unique connection to the origin server.

The caching of layered encoded video is studied in [10, 11]. In [10] an optimization algorithm determines which videos and which layers should be stored in the cache. In [11] the focus is on the maximization of the perceived quality for popular videos that are delivered over best-effort networks.

Unlike traditional web caching, most of the above video caching schemes, do not take into consideration the dynamic nature of caching according to which cache contents are dynamically updated based on demand, but rather employ replication of parts of the videos.

The work that is more closely related to ours is [12]. In that work, each video is segmented into exponentially-sized segments and replacement decisions are applied on such segments. Segmentation is associated with a replacement algorithm which depends on (a) the object reference frequency and (b) the distance of each segment from the beginning of the video.

The work in this paper differs from the work in [12] in numerous ways. First, it proposes two generalized segmentation schemes for the determination of the size of the segments. Second, video segmentation is examined in isolation and it is not associated with a specialized replacement algorithm. This allows for the derivation of conclusions that depend solely on the segmentation scheme irrespectively of the replacement policy. Third, our work considers additional aspects of performance such as the responsiveness to popularity changes and studies the trade-off between responsiveness and byte hit ratio (BHR). Additionally, it considers the case of different access costs for each video and proposes ways to integrate this information into the caching policy. Finally, it provides for a performance comparison of the proposed schemes.

In this work two generalized segmentation techniques are studied and compared with each other: the Fixed Chunk Size segmentation scheme, which gives the opportunity for a thorough understand-

ing of the trade-off between BHR and responsiveness to changes in popularity; the Variable Chunk Size segmentation scheme, which is shown to be capable of combining a small response time with high BHR. To improve the performance of the Fixed Chunk Size segmentation scheme, a variation is proposed which switches between different fixed chunk sizes. Video segmentation is also considered as a mechanism to provide for caching differentiation based on access costs. By employing access cost dependent chunk sizes an overall access cost reduction is achieved. Part of the work presented here was first introduced in [13].

The rest of the work is organized as follows. The basic idea of segment-based caching is motivated in section 2. In section 3 the proxy server architecture is presented and the proposed caching techniques are introduced. The performance of the proposed schemes is evaluated via simulation in Section 4. The work is concluded in Section 5.

2 Motivation of the Work

Cache replacement algorithms utilize the request history to estimate the current request probabilities and self-organize accordingly. Given a time invariant request pattern and a large number of request samples, the underlying request probabilities can be "learned" by counting the requests for each video. Replacement algorithms are able to provide a good estimate of the request probability without the need to count a large number of requests, e.g. the Least Recently Used (LRU) replacement algorithm simply replaces the least recently used object upon a request of a new object.

In web caching the arrival of a single request changes slightly both the recent request history and the state of the cache, since the size of an ordinary web page is very small compared to the capacity of the cache. In video caching a single replacement causes a relatively greater change to the state of the cache, although a single request has similar impact on the recent request history as in web caching. Segment-based replacement strategies (as studied here) try to establish a "Web-like" relation between a single request and its impact on the state of the cache.

The potential advantage of a chunk-based replacement strategy is demonstrated in the following simplified example. Assume that there are two equally sized videos, competing for a place in the

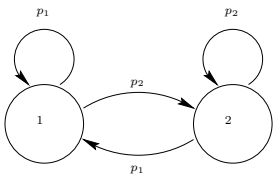

Figure 1: State diagram of the cache for the first scenario(*).

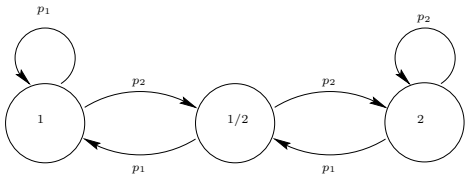

Figure 2: State diagram of the cache for the second scenario(*).

(*)State 1 (state 2) corresponds to video 1 (video 2) being entirely cached and state 1/2 corresponds to the first half of each video being cached. The transition probabilities p_1 and p_2 are equal to the corresponding request probabilities.

cache that can store only one video. Let p_1 (p_2) denote the probability that video 1 (video 2) is requested. In addition assume that the following simple request-based replacement algorithm is used: A video that is not found in the cache upon request, is cached when it arrives from the server, taking up the storage space that was previously occupied by the other video. Each transmission from the server incurs a bandwidth cost equal to the size of the transmitted object. Two scenarios are considered. In the first scenario the replacement unit is equal to the entire video, implying that videos are cached or removed from the cache entirely. The state of the cache upon replacement epochs[1] can be modelled as a two state Markov chain (depicted in Fig. 1). The second scenario allows the partial replacement of video, with a replacement unit that is equal to half of a video. When the requested video is completely or partially missing from the cache, half of the previously cached video is flushed, and half of the requested video is being cached. Replacement takes place in such a way that when a video is partially cached, is is always its initial part that is stored in the cache. This implies that the cache (when full) can be in one of either three states: video 1 fully cached; video 2 fully cached; initial parts of both videos cached. The state diagram of the cache content according to the second scenario is illustrated in Fig. 2.

For both scenarios, let the cost of a total cache miss (the entire requested video is missing from the cache) be equal to 1 and the cost of a partial cache miss (one half of the requested video is missing) be 0.5. If the requested video is completely cached, a cache hit occurs, and no cost is incurred. It is straightforward to calculate the steady state probabilities and costs for each scenario.

[1]We assume that a video is immediately downloaded upon its request, so replacement decisions occur at request arrival instants.

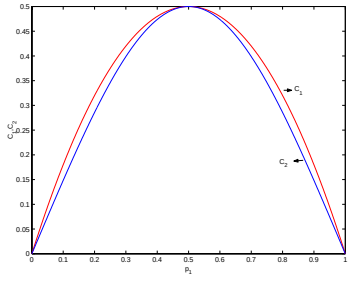

Figure 3: Steady state cost for the two scenarios as a function of the request probability of video 1

More specifically, the steady state cost is equal to $\sum_{i,j} \pi_i P_{ij} c_{ij}$ where P_{ij} is the transition probability from state i to state j, c_{ij} is the cost corresponding to this transition and π_i is the steady state probability of state i. Let C_1, C_2 denotes the steady state cost for the first and the second scenario, then:

$$C_1 = 2p_1p_2 \qquad C_2 = \frac{3p_1p_2}{2(1-p_1p_2)} \tag{1}$$

Fig 3 shows the two costs. It can be seen that $C_1 \geq C_2$ for all p_1 and p_2 (equality holds only for the special cases $p_1 = 0$, $p_1 = 1$, and $p_1 = .5$)[2], meaning that a chunk-based replacement scheme may yield a reduction of cost even for this simplistic system.

3 Segment-Based Caching

3.1 System Architecture

Figure 4, illustrates the topology of the video distribution system under consideration. Videos are stored at geographically dispersed origin servers. A proxy server is installed at the same local area network with a number of clients. Requests for videos are directed to the proxy which services them either from its local cache or by contacting the origin servers over the wide area network. It is assumed that there is abundant bandwidth between the proxy and the clients to support video streaming. On the other hand, the transmission of videos from the origin servers over the wide area

[2]The aforementioned analysis was carried out under the assumption that request interarrivals are always greater than the time it takes to download half or the entire video; this in essence means that replacement decisions are implemented instantly and do not have to wait until the missing data arrive from the origin server.

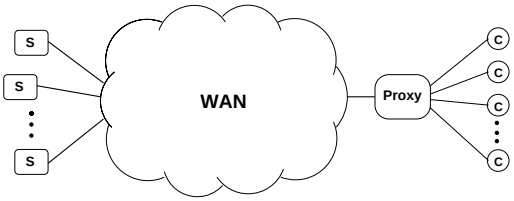

Figure 4: Network topology: the origin servers (S) hold the available videos; clients (C) request videos and the proxy server services these request.

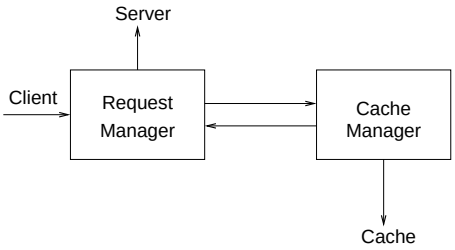

Figure 5: The internal proxy architecture.

network is expensive as it consumes bandwidth, which is a scarce resource in the backbone. The proxy caches the videos trying to improve the delivery delay and also to reduce the access to the servers and consequently the volume of data transmitted over the wide area network.

Figure 5 illustrates the internal architecture of a proxy server. It consists of two major entities: the *request manager* and the *cache manager*. The request manager is responsible for the continuous streaming of video towards the clients. In general, its responsibility is to schedule the transmission of the prefix to the clients and forward a request for the suffix to the origin server. Special is taken to ensure the uninterrupted flow of data to the clients. This in general means that adequate buffers are used to smooth out the delay jitter in both connections (server-proxy and proxy-clients). The main responsibility of the cache manager is to efficiently allocate the proxy's storage resources to the requested videos. This work focuses on the functionality of the cache manager. To allow for the isolated study of the cache manager, only a simple request manager is considered. It is assumed that the request manager immediately initiates the transmission of the prefix (if any) to the client and requests the suffix from the origin server.

The cache manager receives incoming data from the origin server and decides whether they will be cached or not. When the decision is to cache the newly retrieved video segments the cache manager also decides (a) how much space to dedicate to the incoming video, (b) which of the missing segments to hold and (c) which data to remove from the cache to free space for the new video segments.

The third decision is in essence the replacement policy. In traditional web-caching schemes, and

in some video caching schemes as well, the functionality of the cache manager is limited to deciding on the replacement actions. These schemes do not consider object segmentation and consequently do not have to decide on the aforementioned issues which are specific to segment-based caching. Instead they cache the entire objects. In contrast, this work focuses on the first two functions ((a) and (b) above) of the cache manager which are performed before the replacement function.

The first function, that is, the decision about the space allocated to each video, in essence controls the replacement granularity and as it will be shown in the sequel, affects significantly the performance of the proxy. The second function of the cache manager, which is the selection of specific segments of a certain size, does not affect the performance in terms of the cache hit ratio since it does not affect the volume of data that is being cached. However it may affect the user perceived quality or the delay of the perceived stream. For example, when layered encoding is assumed, the selected segments can be specific layers of the video and this selection may affect the perceived quality (analysis, rate). Otherwise, these segments can be consecutive frames at any distance from the beginning of the video and this selection may affect the delay (initial delay, or delay due to VCR operations) experienced by the users. In this work the initial consecutive segments of a video are given preferential treatment, as compared with any other combination of same size data, in order to reduce the initial delay, as neither layered encoding nor VCR operations are considered.

3.2 The Proposed Video Segmentation Schemes

The aforementioned functions that make up the operation of the cache manager, are performed sequentially as follows. The cache manager uses a replacement unit, called a chunk. When a video that is not in the cache is requested it is fetched from the server and its initial segments, up to the size of a chunk are stored in the proxy. In case there is not sufficient space in the cache the replacement algorithm selects a video for removal. The last chunk of the selected video is removed. Each additional request for the same video results in the caching of an additional (consecutive) chunk. This guarantees that only the prefix (initial consecutive parts) of each video is cached.

The size of the chunk determines the granularity of the replacement procedure, and thus, has a

great impact on the overall performance of the cache. In the sequel two video segmentation schemes are presented that are responsible for an appropriate selection of the chunk size: the Fixed Chunk Size and the Variable Chunk Size segmentation schemes.

3.2.1 The Fixed Chunk Size scheme.

The Fixed Chunk Size (FCS) segmentation scheme has the advantage of being particularly simple thus, allowing for a clear exposition of the impact of the replacement granularity on video proxies performance as well as on the underlying trade-off between BHR and adaptability to changes of popularity. Under FCS the size of the chunk is a tunable parameter which can vary from a few frames to the complete video but once it has been selected, it remains the same for all videos and for all requests. As mentioned earlier each request for a video results in the caching of an additional chunk; when the missing part of a requested video is smaller than the chunk size, all the missing segments are kept in the cache. When the size of the chunk is equal to a complete video this scheme reduces to a standard web-like caching scheme.

3.2.2 The Variable Chunk Size scheme

Under the Variable Chunk Size (VCS) scheme, the size of the chunk is a function of the cached part of each video at the time the request is made. This means that (a) typically the chunk size is different for each video and (b) the size of the chunk is not the same for all the requests for a given video. Specifically, the size of the chunk for a requested video i, depends on k_i, the number of chunks of i already in the cache at the time the request was made, through: $chunk(k_i) = g \cdot k_i$, $k_i \neq 0$. The factor g, $g > 0$, will be referred to as the *acceleration factor*. If the requested video is missing from the cache, that is $k_i = 0$, then a few segments are cached as the first chunk, $chunk(0)$. For subsequent requests the size of the chunk is equal to a multiple of the cached segments of video i at the time instant that the request was made. The motivation under this scheme is (a) to provide large chunk size to popular objects and consequently to increase the probability that a greater portion of these objects will be held in the cache and (b) to give the opportunity to new popular videos that are missing from the cache to increase their cached portion faster (within a

fewer requests) by increasing the chunk size in each request.

3.3 Chunk Size Determination Based on the Access Costs

Replacement algorithms such as Least Recently Used (LRU) or Least Frequently Used (LFU), base the replacement decisions solely on objects' popularity. In most practical cases however, there are additional issues that prompt for a differentiation of the treatment offered to each video. Different link costs, may be considered to reflect diverse distance and/or congestion level of a remote server. Content providers that pay for a preferential treatment of their content is another example for treatment differentiation. In these cases there is a need for the caching system to be able to take such factors into consideration. In the present work, the case of a different access cost for each video is considered. In that case the quantity that regulates the decision of the algorithm is a generalized notion of cost which is given by the product of the request probability and the access cost for the requested video. In the sequel two alternative approaches to accommodate this issue are proposed.

The first approach provides differentiation based on the access costs by properly regulating the chunk sizes. For this purpose a video segmentation scheme referred to as *Chunk Based Differentiation* is proposed which is shown to reduce the overall access cost. Under this scheme, an appropriate selection of the chunk size for each video is made based on its access cost. In particular, the largest possible chunk size (a complete video) is assigned to the object with the highest access cost and proportionally (based on the access cost)smaller chunk sizes for the remaining objects. In a sense, this scheme is a combination of the FCS and VCS schemes proposed earlier, as the size of the chunk is different for each video (it is proportional to its access cost) but it is the same for all requests for the same video.

The second approach uses the FCS scheme combined with *probabilistic LRU* and leaves the replacement algorithm to account for the different access costs. Probabilistic LRU has been proposed for web-caching (caching of entire objects) in [14]. Under that scheme, when a new request for a video arrives the least recently requested item is removed from the cache with a probability which is proportional to its access cost and it remains in the cache (no replacement takes place) with the

supplementary probability. The two approaches are compared with each other in section 4 and the benefits of each scheme are presented.

4 Performance Study

4.1 Performance Metrics

As mentioned in Sect. 3.1, the main responsibility of the cache manager is to efficiently manage the proxy's storage resources so as to reduce the volume of the data that are fetched from the origin servers. This performance aspect is captured by the Byte Hit Ratio (BHR), which is the fraction of data that can be served directly from the local storage of the cache. In systems where partial caching is applied a hit in most cases is partial and should capture the portion of data currently on the proxy. Consequently, the BHR for a single request for video i is defined as:

$$BHR_i = \frac{\text{Size of the cached portion of the requested video } i}{\text{Size of the complete video } i}$$

BHR_i takes values between 0 and 1; 0 for a complete miss, and 1 for a complete hit. The average BHR of all requested videos over an interval x is denoted as $BHR(x)$; the interval x can be a time interval (e.g., a day) or a number of requests. The steady state BHR (ss-BHR) is determined from $BHR(x)$ as x tends to infinity assuming that the underlying request process is time invariant (i.e., the probability of a video remains unchanged over the interval x).

4.2 Simulation Model

The request distribution of the videos is assumed to be a zipf-like distribution similar to that observed for web pages [15]. Typically, the request pattern in a proxy server exhibits temporal locality. However due to the lack of access traces from real video servers, and the difficulty of producing a synthetic workload that does exhibit temporal locality while the video popularity follows zipf's law [11, 16], the independent reference (IR) model is assumed. According to the IR model a video i is requested with probability p_i independently of previous requests. The fact that several results obtained with real traces are qualitatively similar to those obtained with the simulation using

Table 1: Simulation parameters

Notation	*System Parameters*	*Default Values*
L	Video Size / Duration	1000 units / 1hour
K	Cache Size	100 videos
N	Number of Videos in the repository	1000
K/N	Relative Cache Size	10%
a	Zipf parameter	0.8
λ	Mean request arrival rate	30 req/hour

the IR model [15] indicates that the IR model is sufficient to predict the relative performance of the proposed caching algorithms.

The simulation model consists of a video server with N videos, and a proxy server with a storage capacity of K complete videos; the ratio K/N captures the relative cache size of the proxy. For simplicity it is assumed that all videos are constant bitrate encoded and have the same length equal to L units. The video length units can be in units of time or storage. p_i denotes the request probability of video i which is also referred to as the popularity of video i. The video popularity follows a Zipf distribution; that is, the request probability for video i is $p_i = C/i^a$, where $C = (\sum_{i=1}^{N} \frac{1}{i^a})^{-1}$ and a is the Zipf parameter determining the skewness of the distribution. Finally, it is assumed that request arrivals follow a Poisson process with mean rate λ. The parameters of the simulation study are summarized in Table 1. The popularity changes considered in the simulation were implemented using the following setting: whenever a popularity change is about to occur, the popularity of videos is transposed i.e., popular videos become unpopular and vise versa. Under the new popularity distribution, unpopular videos that were missing from the cache appear as new hot videos and eventually capture a significant part of the cache. Previously popular videos are made unpopular and are eventually pushed out of the cache.

In the simulations the LRU replacement policy is used with a slight modification that prevents the replacement of chunks belonging to "active videos" (videos that are currently streamed), i.e.,

the least recently used inactive video is selected for the replacement. LRU is chosen as it is the most widely studied replacement policy and is often used as a comparison standard.

4.3 Optimal Static Policy

If the request distribution is a-priori known, and time invariant, then an optimal replication strategy can be derived with the ability to yield a smaller cost than any other conceivable replacement algorithm[3]. Assuming a unicast-only backbone in the path from the server to the proxy, it can be shown that the optimal caching policy – in terms of bytes that cross the backbone link – is the Highest Popularity First (HPF) policy. Under HPF, the proxy stores entire videos in descending order of popularity, until its cache capacity is reached. Only the last video is partially cached. The optimality of HPF stems from the fact that HPF is an optimal solution to the partial knapsack problem: maximize $\sum_{i=1}^{N} v_i \cdot p_i$ under the constraints: $\sum_{i=1}^{N} v_i \leq S$, $0 \leq v_i \leq L_i$, where L_i is the length of video i in number of chunks, v_i is the cached prefix of video i in number of chunks, S is the proxy's storage capacity in number of chunks, and N is the number of available videos (see [6] for details). However in most cases the popularity of the requested objects is unknown and time variant, thus, cache replacement algorithms take over the responsibility to dynamically conform cache contents based on demand.

4.4 Simulation Results

4.4.1 The trade-off between BHR and responsiveness

Figures 6 and 7 provide a visualization of the contents of a cache that employs the FCS scheme and the LRU replacement policy with chunk sizes of 2 and 100 units respectively, as time evolves. These figures are compared with Fig. 8 which depicts the optimal static allocation given by the HPF rule (assuming known request probability). For illustration purposes, only a total population of five videos and a cache capacity of three videos is considered. The cache is empty prior to the first

[3]The optimal static policy is presented here, as it will be a point of reference in the sequel, even though the assumption of a known and time invariant environment is not the one under which the proposed segment-based replacement algorithms were designed.

request arrival and fills up as requests arrive. The LRU replacement policy is activated as soon as the total capacity is reached. The video size is assumed to be 1000 units. The request distribution is assumed to follow the Zipf's law with parameter $a = 0.8$ for video-1 to video-5 (descending popularity). From Fig. 6 and Fig. 7 it becomes clear that for a small chunk size, equal to 2 units, the cache state converges to the optimal static allocation slowly and with negligible oscillations, while for a larger chunk size, equal to 100 units (1/10 of the video length), the cache state converges fast but significant oscillations appear. From this results the underlying trade-off between the BHR and the duration of the convergence to the ss-BHR becomes visible. Oscillations are expected to have a negative impact on the BHR while the convergence time is expected to affect the capability of the system to adapt to popularity changes. This trade-off is investigated in detail in the sequel.

4.4.2 The effect of the relative cache size and the request distribution

Fig. 9 illustrates the effect of the relative cache size on ss-BHR, for several chunk sizes. As expected the BHR increases with the relative cache size since a greater number of chunks fit in the cache. Similar observations apply to Fig. 10 that depicts the ss-BHR as a function of the Zipf parameter. The VCS scheme outperforms FCS especially for small caches sizes and highly skewed request distributions. This results are reasonable since popular objects are treated preferentially by the VCS scheme which allocates to them a larger chunk size resulting in a larger, effective occupation of the cache storage capacity.

Figures 11 and 12 (for chunk sizes 10 and 1000 respectively) depict the BHR versus the sum of

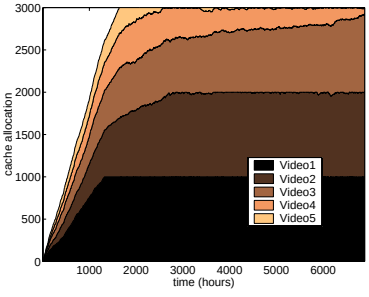

Figure 6: Cache state for chunk size = 2 units

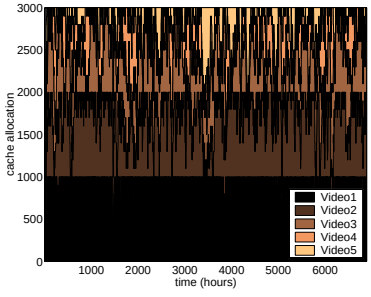

Figure 7: Cache state for chunk size = 100 units

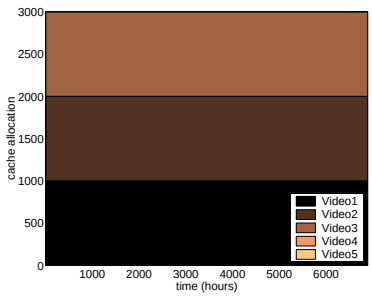

Figure 8: Optimal static allocation

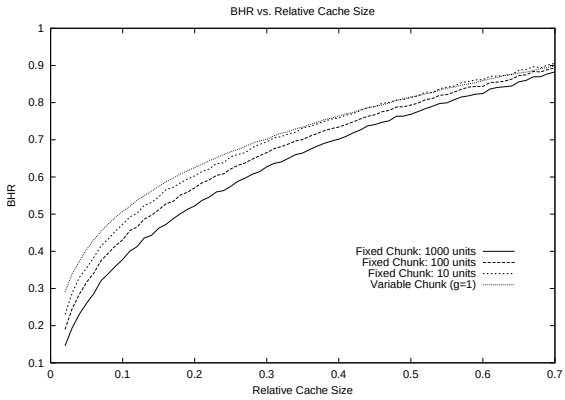

Figure 9: The ss-BHR against the relative cache size for several chunk sizes

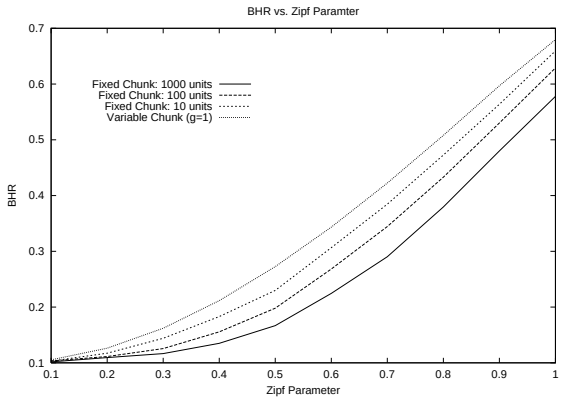

Figure 10: The ss-BHR against the Zipf parameter a, for several chunk sizes

the popularities $\sum_{i=1}^{m} p_i$ of the videos that fit in the cache[4], when videos are placed in the cache in descending order of popularity; m is the index of the least popular video that fits in the cache. This sum depends on two factors: the size of the cache and the skewness of the popularity distribution[5]. Each line in Figures 11 and 12 corresponds to a different value of the Zipf parameter and the points of each line correspond to different values of the cache size. From the figures it follows that for a small chunk size different pairs of skewness and cache size result in the same BHR if the sum of the popularities of the videos that fit in the cache is the same. That is, the latter sum fully determines BHR under a small replacement unit. This conclusion suggests that a smaller cache size would be required to achieve a certain BHR if the video request probabilities are highly skewed, compared to the case under less skewed request probabilities. For a greater chunk size this result seems to hold only for zipf parameters greater than some value e.g 0.5.

4.4.3 The effect of the chunk size

Fig. 13 illustrates the ss-BHR as a function of the chunk size for the FCS scheme, under a non-changing popularity distribution and for the system parameters presented in Table 1. It is observed

[4]These figures relate to a discussion that is motivated by results in [18], where the relation between the fault probability of LRU and the tail of the popularity (request) distribution is demonstrated.

[5]For a specific cache size, this sum increases as the zipf parameter increases, since a greater request probability (p_i) is associated with the videos involved in the sum. For a specific value of the zipf parameter the sum increases as the cache size increases, since more videos are involved in the summation.

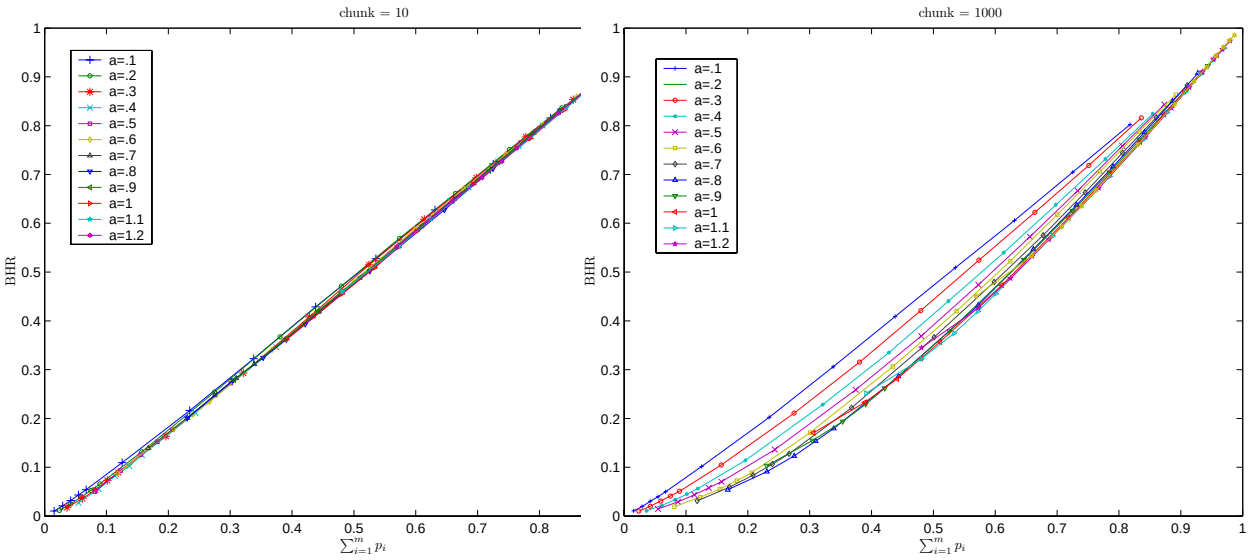

Figure 11: ss-BHR vs $\sum_{i=1}^{m} p_i$ for FCS with chunk size: 10, where m is the index of the least popular video that fits in the cache

Figure 12: ss-BHR vs $\sum_{i=1}^{m} p_i$ for FCS with chunk size:1000, where m is the index of the least popular video that fits in the cache

that as the chunk size increases, the BHR reduces initially fast and then slowly converges to a value which is the steady state BHR of a web-like video caching scheme. Similar observations apply to Fig. 14 which illustrates the ss-BHR as a function of the acceleration factor for the VCS scheme. Generally a large value of the acceleration factor leads to large chunk sizes and consequently to the reduction of the BHR. Observe that the impact of the acceleration factor in the VCS scheme, is rather small comparatively with the impact of the chunk size on the FCS scheme (note the different scale in the axis of these two figures) .

In the results presented above it was assumed that the popularity of videos does not change, that is, the underlying request probability is time invariant. In a more realistic scenario, where the demand distribution changes, the BHR is expected to decrease for some period and then to converge to the new steady-state value. It should be noted that the new ss-BHR is not necessarily the same with the old one as it depends on the skewness of the new popularity distribution. Responsiveness can be qualitatively defined as the ability of the system to adapt to changes in popularities. In order to capture this performance aspect, we flush the cache[6] and measure the time needed for the

[6]This is an extreme case of popularity change since it is equivalent to a cache full with totally unpopular videos

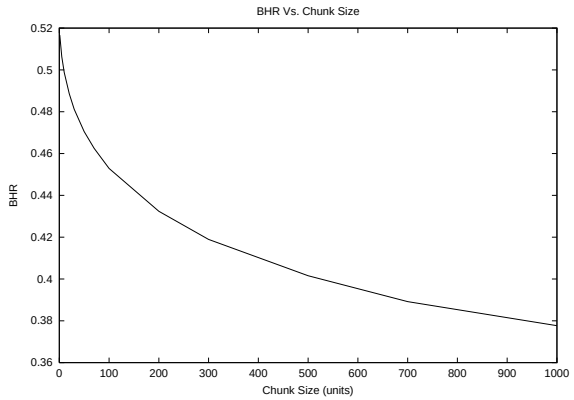

Figure 13: The ss-BHR for different chunk sizes, for the FCS scheme.

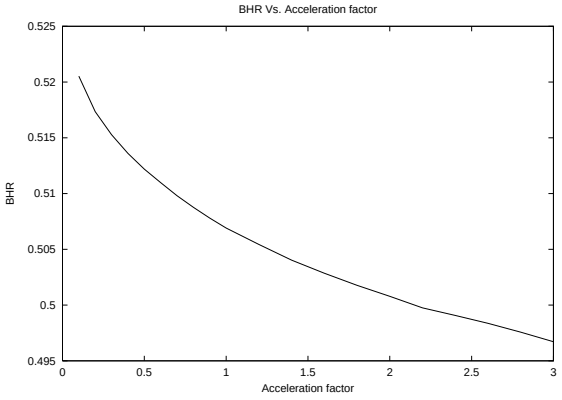

Figure 14: The ss-BHR for different values of the acceleration factor, for the VCS scheme.

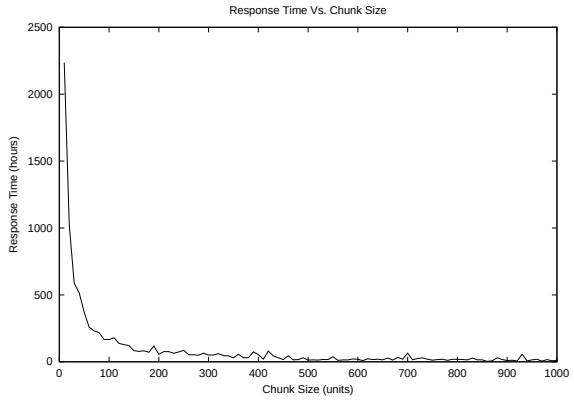

Figure 15: Response time for the FCS scheme

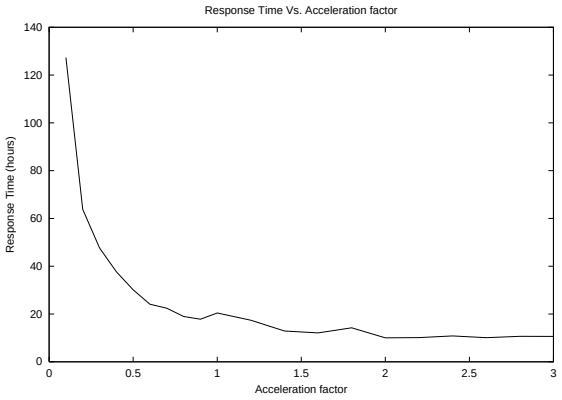

Figure 16: Response time for the VCS scheme

BHR to reach 90% of its steady-state value. Figures 15 and Fig. 16 illustrate the response time as a function of the chunk size and the acceleration factor, for FCS and VCS schemes respectively. For the FCS scheme, the response time is smaller for large chunk sizes and increases rapidly as the chunk size decreases. The same happens in the VCS scheme with the acceleration factor. What is worth noting in these figures is the relatively small response time under the VCS scheme even for very small values of the acceleration factor.

Comparing figures 13, 15 with 14, 16 it can be concluded that the VCS scheme achieves a more effective compromise in the trade-off between BHR performance and responsiveness compared to the FCS scheme. For example, a BHR of 50% can be achieved with a fixed chunk size of 10 units (Fig. 13) at the cost of a huge response time of 1000 hours (Fig. 15). The same BHR can be achieved under the proposed VCS scheme with accelerator factor 2 (Fig. 14) at the cost of response

time of 10 hours (Fig. 16).

4.4.4 The BHR under popularity changes

The responsiveness of the caching system affects significantly the overall performance especially when demand changes occur frequently. In such cases, the ss-BHR may practically never be reached if the caching system has a long adaptation period compared to the frequency that changes occur. This could lead a system that appears to have a high ss-BHR to a worst long term average BHR since transient periods dominate the long term performance. This is illustrated in Figures 17 and 18. These figures show the BHR(24h) versus time, under the FCS scheme, for different chunk sizes[7]. The BHR initially oscillates around the steady-state value; In periods where popularity remains stable a smaller chunk provides for a better BHR. For the periods that follow a popularity change the BHR reduces for both chunk sizes but the reduction is smaller under a large chunk, as this allows for a quick adaptation to the new popularity (observe the shallow gap in Fig. 17. A small chunk size, although performing better under static popularity, it is outperformed during periods of popularity changes as it needs a considerably larger time to catch-up with the new demand changes and consequently, the gaps are deeper. The average BHR over the entire observation window, is equal to 0.45 for the system that uses a chunk size of 50 units and 0.425 for the system that uses a chunk size of 200 units. On the other hand, in Fig. 18 where demand changes occur more often (every 170 hours), the system that uses a chunk size of 200 units achieves higher average BHR (0.41 over 0.39).

In figures 19 and 20 a comparison of the VCS and FCS scheme is illustrated. Fig. 19 shows the BHR(24h) versus time for the FCS with chunk size of 30 units and the VCS scheme with $g = 1$[8]. The BHR initially oscillates around the steady-state value for both schemes. At time instant 500 a demand change occurs. In the period that follows this change, the BHR reduces for both schemes

[7]In order to achieve a fast convergence to the steady state, the cache is considered initially full with the most popular videos that fit in the cache.

[8]In order to achieve a fast convergence to the steady state, the cache is considered initially full with the most popular videos that fit in the cache.

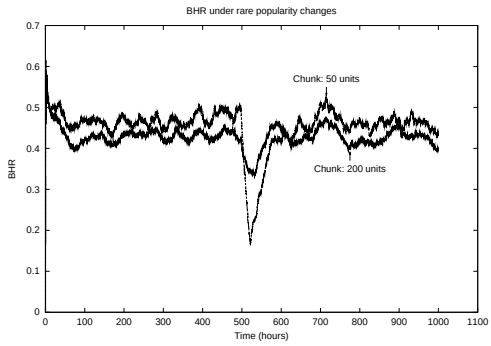

Figure 17: Rare popularity changes. Average BHR for chunk 50: 0.45. Average BHR for chunk 200: 0.425.

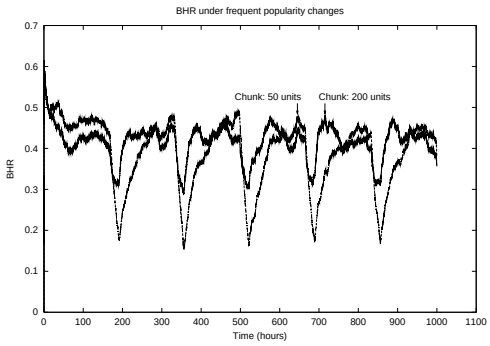

Figure 18: Frequent popularity changes. Average BHR for chunk 50: 0.39. Average BHR for chunk 200: 0.41.

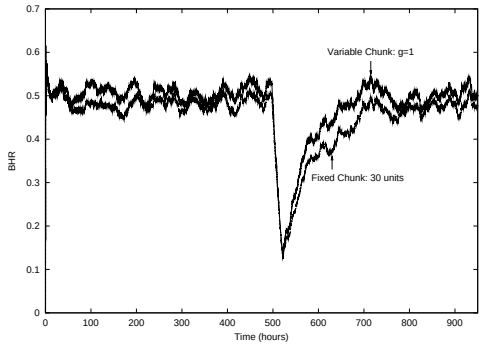

Figure 19: The BHR(24h) versus time for the FCS with chunk size of 30 units and VCS with g = 1. A sudden change in the demand pattern occures at time instant 500.

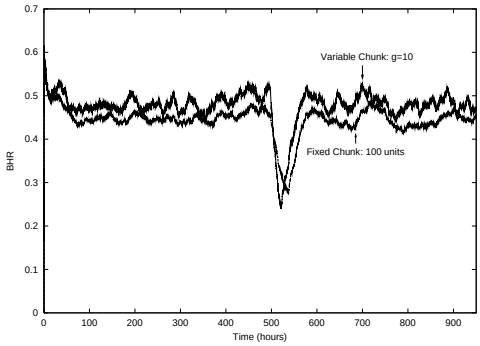

Figure 20: The BHR(24h) versus time for the FCS with chunk size of 100 units and VCS with g = 10. A sudden change in the demand pattern occures at time instant 500.

and then again converges to the steady state value. It is observed that the VCS scheme with $g = 1$ achieves higher BHR than FCS with chunk size of 30 units on both the steady-state and the transient period, resulting to a higher long term average BHR.

Similar observations apply to Fig 20 where the VCS scheme with $g = 10$ is compared with the FCS with chunk size of 100 units. These results indicate that for any fixed chunk size of the FCS scheme, an appropriate value of the acceleration factor (g) of the VCS may be identified such that the VCS scheme results to a higher BHR than the FCS scheme in both the steady-state and the transient period, achieving a higher long term average BHR.

4.4.5 Dynamic Adaptation of the Chunk Size

In this section, a variation of the FCS is presented which improves the performance of pure FCS scheme by switching between different chunk sizes at different time periods. The main idea is to

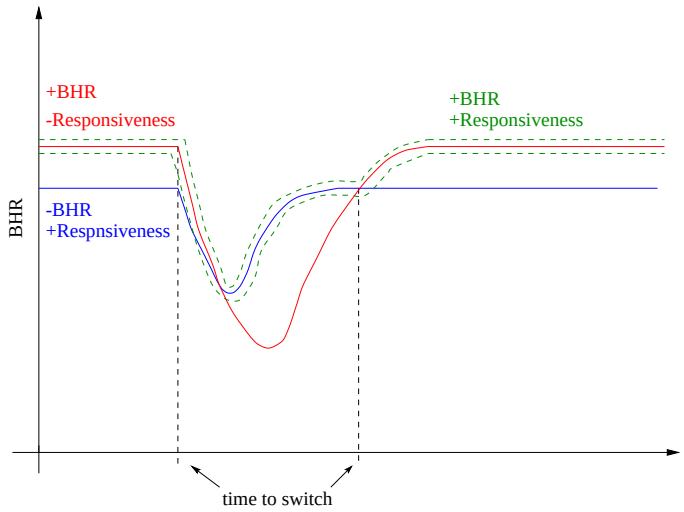

Figure 21: Dynamic adaptation of the chunk size; Main idea

use a small chunk size for periods where the popularity remains stable, thus achieving a high BHR during these periods, and switch to a larger chunk size during periods where a change of popularity occurs. Sophisticated methods can detect the change of popularity by looking for sudden decreases of BHR and adjusting the chunk size automatically. In any case, once the cache content has been updated the system could switch back to a small chunk size in order to achieve higher ss-BHR. This is idea is illustrated in Fig 21.

The benefits under a dynamic selection of the chunk size are illustrated in Fig. 22 and Fig. 23 where the BHR is depicted as a function of time. The parameters are the same as those in Fig. 17 and Fig. 18 respectively. From these figures it becomes clear that a change of the chunk size at the right moment combines the advantages of both, small and large, chunk sizes and results in a better overall BHR performance than under the corresponding static schemes. Note that the average BHR under the dynamic chunk selection (0.46 (0.44) for the system of Fig. 22 (Fig. 23)) is higher than that under the fixed chunk size schemes for chunk size 50 (0.45 (0.39)) and chunk size 200 (0.425 (0.41)). The application of dynamic chunk selection corresponding to Fig. 17 is illustrated in Fig. 22.

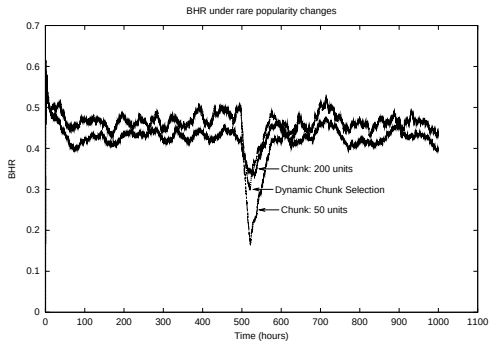

Figure 22: Rare popularity changes & Dynamic Chunk Selection. Average BHR for chunk 50: 0.45. Average BHR for chunk 200: 0.425. Average BHR for dynamic chunk selection: 0.46.

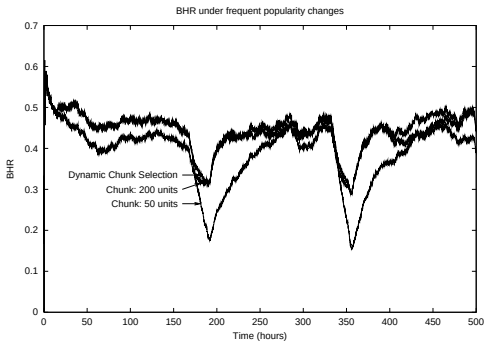

Figure 23: Frequent popularity changes & Dynamic Chunk Selection. Average BHR for chunk 50: 0.39. Average BHR for chunk 200: 0.41. Average BHR for dynamic chunk selection: 0.44.

4.4.6 The Chunk Based Differentiation scheme

In this section some results are presented illustrating the ability of the Chunk Based Differentiation scheme to reduce the overall cost when a different access cost for each video is assumed following the arguments of section 3.3. In order for the results to be comparable with those in the previous section it is assumed that all objects are requested with probability $p_i = 1/N$ and that the access costs follow a zipf-like distribution similar to the popularity distribution in the previous section. Equivalently with the BHR the performance metric of interest that is considered is the *cost reduction ratio*

$$CRR = \frac{\sum\limits_{i \in R} k_i \cdot c_i}{\sum\limits_{i \in R} L_i \cdot c_i}$$

where, R is the set of video indexes which correspond to the requests made to the proxy and k_i, c_i and L_i are the size of the cached portion, the access cost, and total size, respectively, of the video associated with request i.

In Fig. 24 the performance of the proposed algorithm for several cache sizes is illustrated. It is observed that although FCS with probabilistic LRU performs better, the general behavior of the Chunk Based Differentiation scheme, is similar to that of FCS with probabilistic LRU when no demand changes occur. However since under the Chunk Based Differentiation scheme the chunk sizes are different for each video, the responsiveness of this scheme will be closer to that of the VCS scheme presented in Fig 16. In contrast, the responsiveness of the FCS with probabilistic LRU will

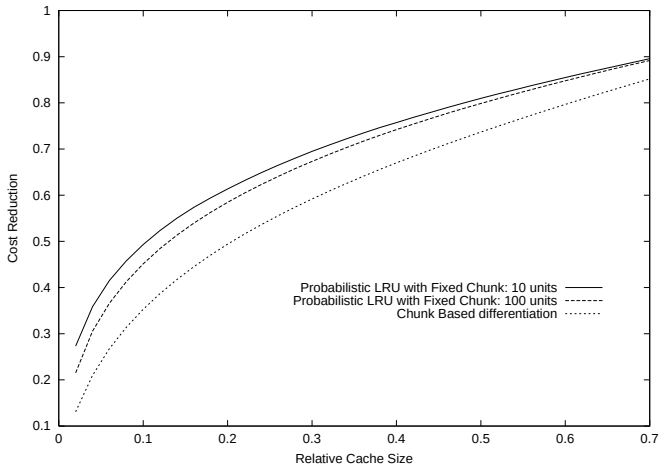

Figure 24: Cost reduction ratio of FCS with probabilistic LRU and Chunk Based Differentiation for several cache sizes

be far worst than that of FCS with plain LRU presented in Fig. 15, since under the probabilistic LRU, only a small fraction of requests is taken into account. Based on the findings about the trade-off between efficiency and adaptability to demand changes discussed in figures 19, 20 and more analytically in [13], the difference in the response time and the relatively small difference in the cost reduction ratio under no demand changes, make the Chunk Based Differentiation scheme an attractive solution when demand changes occur.

5 Conclusions

In this paper, several schemes for caching large media objects have been presented. For such large objects the entire object caching has been shown to be inefficient mainly due to the fact that storing only a few objects can easily exhaust the proxy storage capacity. As a solution to this problem it is proposed that large media objects be segmented into a number of chunks and replacement decisions be taken at the chunk level rather than based on the entire objects. It has been shown that the size of the chunk, that is the replacement granularity, significantly affects the performance of the proxy. Two video segmentation schemes have been proposed. The study of the Fixed Chunk Size segmentation scheme clearly reveals the trade-off between BHR and responsiveness but provides poor performance in terms of the long term average BHR, a performance metric which combines both the BHR and the responsiveness to demand changes. The introduced Variable Chunk Size

segmentation scheme is shown to be capable of combining a small response time with high BHR, thus resulting to a higher long term average BHR, by adjusting the size of chunk based on object access frequencies. Moreover, a variation of the fixed chunk size segmentation scheme is proposed and shown to improve its performance by switching between different chunk sizes.

The benefit of segment-based caching is also demonstrated for the case where different access cost for each video is considered. Two schemes have been considered: one that combines probabilistic LRU and fixed chunk sizes and another in which the chunk size is variable and based on the access cost for each video. Under a fixed demand distribution it is shown that the former scheme achieves a higher cost reduction than the Chunk Based Differentiation scheme. Nevertheless it is argued that due to its good cost reduction ratio and better response to demand changes the later scheme may be particularly effective under a demand changing environment in which the FCS scheme and the probabilistic LRU respond rather slowly.

References

[1] Scott A. Barnett, Gary J. Anido, and H.W. Beadle, "Caching policies in a distributed video on-demand system," in *Australian Telecommunication Networks and Applications Conference*, Sydney, Australia, 1995.

[2] J.-P. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz, "Networking requirements for interactive video on demand," *IEEE Journal on Selected Areas in Communications*, vol. 13,5, pp. 779–787, 1995.

[3] Christos Papadimitriou, Srinivas Ramanathan, P Venkat Rangan, and Srihari Sampathkumar, "Multimedia information caching for personalized video-on-demand," *Computer Communications*, vol. 18, no. 3, Mar. 1995.

[4] Subhabrata Sen, Jennifer Rexford, and Don Towsley, "Proxy prefix caching for multimedia streams," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.

[5] Zhi-Li Zhang, Yuewei Wang, D. H. C. Du, and Dongli Su, "Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, Aug. 2000.

[6] Bing Wang, Subhabrata Sen, Micah Adler, and Don Towsley, "Proxy-based distribution of streaming video over unicast/multicast connections," Technical Report UMASS TR-2001-05, University of Massachusetts, Amherst, 2001.

[7] S. Ramesh, I. Rhee, and K. Guo, "Multicast with cache (mcache): An adaptive zero-delay video-on-demand service," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Anchorage, Alaska, Apr. 2001.

[8] S.-H. Gary Chan and Fouad A. Tobagi, "Caching schemes for distributed video services," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*, Vancouver, Canada, June 1999.

[9] Markus Hofmann, T.S. Eugene Ng, Katherine Guo, Paul Sanjoy, and Hui Zhang, "Caching techniques for streaming multimedia over the internet," Tech. Rep., Bell Laboratories, May 1999.

[10] Jussi Kangasharju, Felix Hartanto, Martin Reisslein, and Keith W. Ross, "Distributing layered encoded video through caches," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Anchorage, Alaska, Apr. 2001.

[11] Reza Rejaie, Haobo Yu, Mark Handley, and Deborah Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Tel Aviv, Israel, Mar. 2000.

[12] Kun-Lung Wu, Philip S. Yu, and Joel L. Wolf, "Segment-based proxy caching of multimedia streams," in *WWW 2001*, Hong Kong, China.

[13] Elias Balafoutis, Antonis Panagakis, Nikolaos Laoutaris, and Ioannis Stavrakakis, "The impact of replacement granularity on video caching," in *Proceedings of IFIP Networking 2002*, Pisa, Italy, May 2002, pp. 214–225.

[14] David Starobinski and David Tse, "Probabilistic methods for web caching," *to appear in Performance Evaluation (Special Issue of SAPM 2000 workshop).*

[15] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, 1999.

[16] Barford Paul and Crovella Mark, "Generating representative web workloads for network and server performance evaluation," in *Measurement and Modeling of Computer Systems*, 1998, pp. 151–160.

[17] Philippe Flajolet, Gardy Danièle, and Thimonier Loys, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229.

[18] Predrag R. Jalenković, "Asymptotic approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities," *Annals of Applied Probability*, vol. 9, no. 2, pp. 430 – 464, 1999.