The Impact of Replacement Granularity on Video Caching

Elias Balafoutis, Antonis Panagakis, Nikolaos Laoutaris, and Ioannis Stavrakakis

Department of Informatics & Telecommunications,
University of Athens, 15784 Athens, Greece
{balaf,apan,laoutaris,istavrak}@di.uoa.gr

Abstract. In this paper the idea that large objects, such as video files, should not be cached or replaced in their entirety, but rather be partitioned in chunks and replacement decisions be applied at the chunk level is examined. It is shown, that a higher byte hit ratio (BHR) can be achieved through partial replacement. The price paid for the improved BHR performance is that the replacement algorithm, e.g. LRU, takes a longer time to induce the steady state BHR. It is demonstrated that this problem could be addressed by a hybrid caching scheme that employs variable sized chunks; the use of small chunks leads to the maximization of BHR in periods of stable video popularity, while large chunks are used when extreme popularity changes occur to assist the fast convergence to the new steady state BHR.

1 Introduction

The explosive growth of demand for bandwidth, fuelled by the introduction of the world wide web in the early nineties, 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, replication of popular data objects close to the demanding 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. The large size of videos not only overloads data connections but also easily exhausts the capacity of conventional web caches.

A variety of caching schemes have been proposed to handle video. Initial works, have inherited the main characteristics of web caching schemes and have treated videos as single entities which are either cached completely or not at all [1,2,3]. More recent works take into consideration the special characteristics of videos : their structure, the associated rate variability, their large volume and the need for a time-constrained delivery of content. The later does not only refer to a requirement for a small initial delay, to preserve the interactivity of the

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 214–225, 2002.

service, but also to a requirement for an isochronous delivery of the media units (video frames) that make up a video stream.

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 in 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, again 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 gap between the prefix and the currently multicasted part of the suffix. Request merging is also proposed in [8,9] in the form of window-based caching schemes. In particular, local proxies cache a sliding window of data, trying to merge requests for the same stream that arrive closely in time.

The caching of layered encoded video is studied in [10,11]. In [10] an optimization algorithm determines which videos and which layers should be cached. 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 (most of them have been designed for video on demand systems) do not conform to the dynamic nature of caching according to which cache contents are dynamically updated in a demand driven fashion.

The performance of a proxy is characterized by its ability to reduce the amount of data that cross the backbone network (captured by the BHR), and also by the ability to provide streaming services with acceptable initial delay. The overall performance is sensitive to sudden changes in popularity. If the cache does not detect such changes quickly, there could be substantial mismatch between the content of the cache and the upcoming content requests, leading to a low hit ratio and an increased initial delay.

In what follows it is proposed that a video be segmented into a number of chunks and replacement decisions be taken at the chunk level rather than based on the entire video. This partial caching has a twofold beneficial effect: it increases the BHR compared to entire video caching; and it reduces the perceived delivery delay, as a significant number of initial parts may be found in the cache upon request. The existence of the initial part of a stream in the cache also allows for jitter concealment in the path from the proxy to the server. Our work studies the effect of the replacement granularity on cache performance and the trade-off that exists between cache performance and responsiveness to popularity changes. We show that the price paid for using partial caching is a slower reaction to popularity changes. We attempt to eleviate this drawback by using variable sized replacement units.

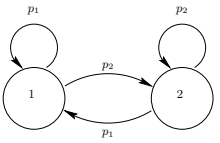

Fig. 1. State diagram of the cache for the first scenario(*).

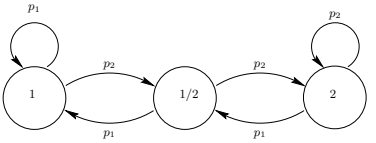

Fig. 2. State diagram of the cache for the second scenario(*).

(*)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.

2 Motivation of the Work – Intuitive Considerations

Cache replacement algorithms utilize the request history to estimate the current request probabilities and self-organize accordingly. Given a stationary 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 ranking without the need to count a large number of requests, e.g. LRU 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. Chunk-based replacement strategies (as studied here) try to establish a "Web-like" relation between a single request and the corresponding replacement unit.

The potential advantage of 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 cache that can fit entirely only one video. Let p_1 (p_2) denote the probability that video 1 (video 2) is requested. In addition assume that a 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 storage space that was held by the other video. 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, one half of the previously cached video is flushed, and one half of the requested video is being cached. Replacement takes place in such a way that if a video is partially

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

cached, the cached part is always its first half. This implies that there are three possible states of a full cache, namely the first half of each video, the entire video 1, or the entire video 2 being 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. 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. If C_1 (C_2) denotes the steady state cost for the first (second) scenario, then:

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

which implies 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].

3 System Description

3.1 Network Topology

Figure 3, illustrates the topology of the video dissemination 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. The proxy caches the most popular videos trying to reduce the accesses to the servers and consequently the volume of data transmitted 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 network is expensive as it consumes bandwidth, which is a scarce resource in the backbone.

3.2 Proxy's Internal Architecture

Figure 4 illustrates the internal architecture of the porxy. The proxy consists of two major entities: the request manager and the cache manager. The request manager accepts all the client requests and is responsible for the continuous streaming of video towards the clients. In general, its responsibility is to schedule

[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.

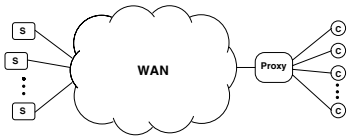

Fig. 3. Network topology: the origin servers (S) hold the available videos; clients (C) request videos and the proxy server services these request.

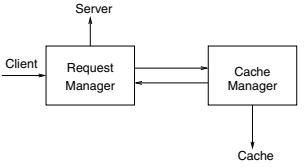

Fig. 4. Internal proxy architecture

both the transmission of the prefix to the clients. The main responsibility of the cache manager it to efficiently allocate the proxy 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. In addition, it is assumed that the suffix can be (and is) delivered exactly when it is needed from the origin server[3].

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

In particular, the cache manager uses a fixed caching/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 chunk is 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 are cached. The objective is to investigate the impact of the replacement granularity on the overall performance of the system.

4 Performance Evaluation

4.1 Preliminaries

As mentioned in Sect. 3.2, 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

[3] A more advanced request manager could, for example, perform request batching to service nearby requests with a single connection to the client.

served directly from the cache's local storage. Here, 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[4] x is denoted as $BHR(x)$. The steady state BHR (ss-BHR) is determined from $BHR(x)$ as x tends to infinity and assuming that no popularity changes occur, i.e. the request probability of a video is assumed to remain unchanged over the interval x.

The independent reference model [12] is assumed, according to which a video i is requested with probability p_i independently of previous requests. If the request probability of each video is known and 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).

The performance of the proposed video caching scheme is evaluated via simulations. The main metric of interest is the BHR. The responsiveness of the cache to popularity changes is jointly considered.

4.2 Simulation Model

We have constructed a simulation model that consists of a video server with a set of 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 are of equal length L units (under the constant bitrate assumption video length units can be time or storage units). p_i denotes the request probability of video i – it is also refered to as the popularity of video i – and follows a Zipf distribution, i.e., $p_i = C/i^a$, where $C = (\sum_{i=1}^{N} \frac{1}{i^a})^{-1}$. a is the Zipf parameter determining the skewness of the distribution. 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, we transpose the popularities of videos, i.e., popular videos become unpopular and vise versa. Under the new popularity distribution, unpopular videos that were

[4] the interval x can be a time interval (e.g., a day) or a number of requests.

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	0.1
a	Zipf parameter	0.8
λ	Mean request arrival rate	30 req/hour

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 our 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 was chosen as the most widely studied replacement policy which is often used as a comparison standard.

5 Simulation Results

Cache State. Figures 5 and 6 provide a visualization of the content of the cache as time evolves. 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 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 results for chunk sizes of 2, and 100 units and a Zipf parameter $a = 0.8$ for video-1 to video-5 (descending popularity) are shown in Fig. 5 and 6 respectively. In Fig. 7 the optimal static allocation of the cache storage is illustrated. From Fig. 5 and Fig. 6 it becomes clear that for a small chunk size the cache state converges to the optimal static allocation slowly and with negligible oscillations, while for a greater chunk size, the cache state converges fast but significant oscillations appear. Oscillations are expected to

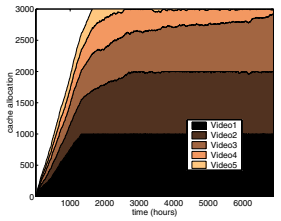

Fig. 5. Cache state for chunk size = 2 units

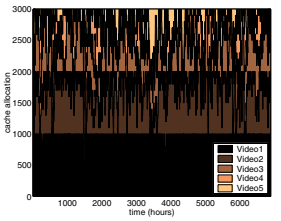

Fig. 6. Cache state for chunk size = 100 units

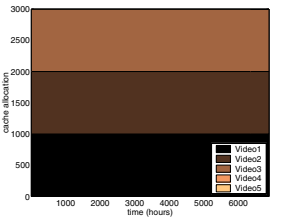

Fig. 7. Optimal static allocation

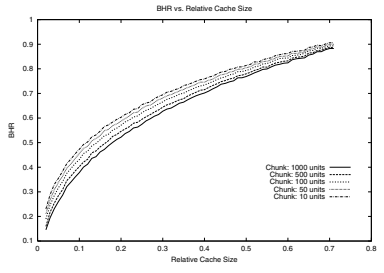

Fig. 8. BHR vs. Relative Cache Size. Zipf parameter 0.8

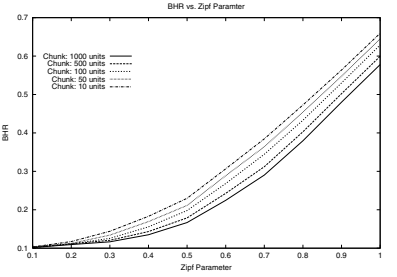

Fig. 9. BHR vs. Zipf parameter. Relative Cache Size 10%

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. The underlying tradeoff is investigated in detail in the sequel.

The Byte Hit Ratio. Fig. 8 illustrates the effect of the relative cache size on BHR, for several chunk sizes. BHR increases with the relative cache size since a greater number of chunks fit in the cache. Moreover, for a specific relative cache size, small chunks lead to a higher BHR. Similar observations apply to Fig. 9 that depicts the BHR as a function of the Zipf parameter, for different values of the chunk size.

Figures 10 and 11 [5] (for chunk sizes 10 and 1000 respectively) depict the BHR versus the sum of the popularities $\sum_{i=1}^{m} p_i$ of the videos that fit in the cache, when videos are placed in the cache according to descending popularity order; 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[6]. Each line in Figures 10 and 11 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.

The impact of the chunk size on BHR is illustrated in Fig. 12, for the system parameters presented in Table 1. For these parameters the sum of the popu-

[5] These figures relate to a discussion that is motivated by results in [13], where the relation between the fault probability of LRU and the tail of the popularity (request) distribution is demonstrated.

[6] 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.

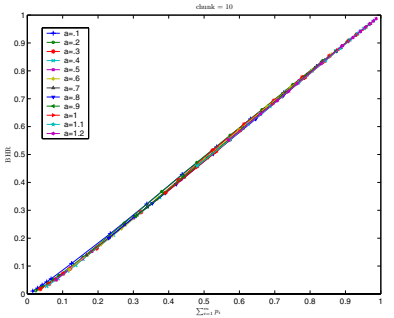

Fig. 10. BHR vs $\sum_{i=1}^{m} p_i$ for chunk = 10, where m is the index of the least popular video that fits in the cache

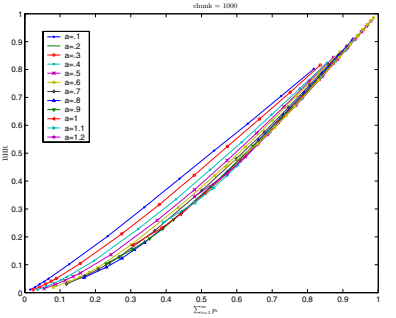

Fig. 11. BHR vs $\sum_{i=1}^{m} p_i$ for chunk = 1000, where m is the index of the least popular video that fits in the cache

larities of the videos that fit in the cache when videos are placed in the cache according to descending popularity order is 0.525. It is observed that as the chunk size increases, the BHR reduces initially fast and then slowly converges to the BHR achieved when complete videos are used as a replacement unit.

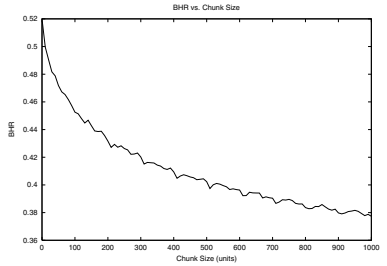

Fig. 12. BHR vs. Chunk Size, for Relative Cache Size 10% and Zipf parameter 0.8

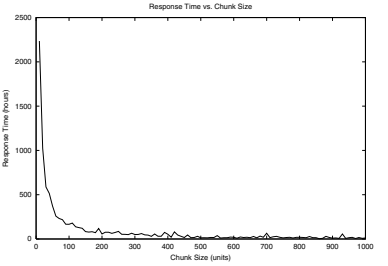

Fig. 13. BHR vs. Response Time, for Relative Cache Size 10% and Zipf parameter 0.8

Responsiveness. Upon a change of popularities, the BHR is expected to decrease for some period and then 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 skweness of the popularity distribution. Responsiveness can be qualitatively defined as the ability of the system to adapt to changes in popularities. In order to quantitatively capture this performance aspect, we flush the cache[7] and measure the time needed for the BHR to reach 90% of its steady-state value. In Fig. 13 the response time is illustrated for several chunk sizes. As expected, the response time is small for large chunk sizes and increases rapidly as the chunk size decreases.

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

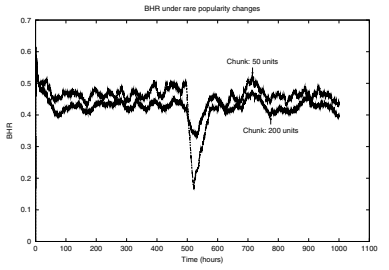

Fig. 14. Rare popularity changes. Average BHR for chunk 50: 0.45. Average BHR for chunk 200: 0.425.

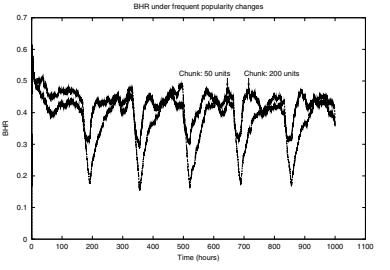

Fig. 15. Frequent popularity changes. Average BHR for chunk 50: 0.39. Average BHR for chunk 200: 0.41.

The effect of popularity changes. From the results presented so far, it is evident that the performance of the system depends not only on the chunk size but also on the frequency of popularity changes (and on how dramatic these changes are). Under a fixed popularity distribution, a small chunk ensures a better steady state BHR than a large chunk size. In practice the BHR under a small chunk size is never reached, as it requires a long adaptation period which is not available under frequent changes in popularity. This could lead to an overall performance that is worse than that achieved under a large chunk size. Figures 14 and 15 illustrate the BHR(24h) versus time for two different cases [8]. Fig. 14, corresponds to the case where demand changes occur rarely (only one at time instant 500). It is observed that in periods where popularity remains stable a smaller chunk provides for better BHR. For the periods that follow a popularity change the BHR reduces for both chunk sizes but the reduction is smaller for a large chunk, which quickly adapts to the new popularity (observe the shallow gap). 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. 15 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).

Adaptation to Changes of Popularity. To cope with sudden changes of popularity it is proposed that the system use a larger chunk size during periods when a change of popularity has just occurred. 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.

[8] In order to achieve a fast convergence to the stready state, the initial state of the cache is considered to coincide with the allocation under HPF

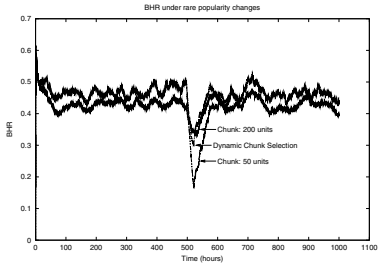

Fig. 16. Rare popularity changes & Dynamic Chunk Selection. Average BHR for chunk 50: 0.45. Average BHR for chunk 200: 0.425. Average BHR for dynamic chunk selection: 0.46.

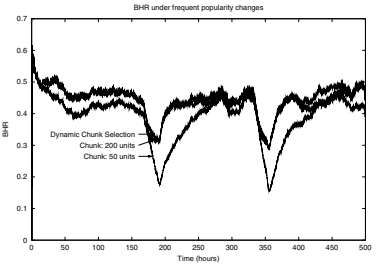

Fig. 17. Frequent popularity changes & Dynamic Chunk Selection. Average BHR for chunk 50: 0.39. Average BHR for chunk 200: 0.41. Average BHR for dynamic chunk selection: 0.44.

The benefits under a dynamic selection of the chunk size are illustrated in Fig. 16 and Fig. 17 where the BHR is depicted as a function of time. The parameters are the same as those in Fig. 14 and Fig. 15 respectively. From these figures it becomes clear that a change of the chunk size at the right moment combines the advantages of both 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. 16 (Fig. 17)) 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. 14 is illustrated in Fig. 16.

6 Additional Considerations

The presented system has taken a rather simplistic approach, as far as the cost of the backbone bandwidth is concerned, by treating all video data as being equal. This has mainly been dictated by our desire not to obscure this first exposition of the effects of partial caching with additional parameters that are not directly related to this issue. In reality not all bits entail the same cost nor should be treated the same way. Some of the reasons for which it may be appropriate to differentiate among videos are:

Server-proxy distance: In general, the greater the distance between the server and the proxy (e.g. in number of hops), the higher the cost of fetching a video from that server. This mean that a cache hit results in a greater reduction of required bandwidth when the requested video belongs to a distant server.

Different link costs: Links may have different costs due to different bandwidth-availability/bandwidth-demand ratios. For example, the proxy could give preferential treatment to content that resides in an origin server that is situated behind a congested link. Loss and/or delay measurements could be used for the estimation of the congestion level.

Content differentiation: Some content could be given preferential treatment as requested by the content provider, e.g., some clips could be given some sort

of priority in the cache, so as to be available upon a request of a popular web page that includes them, with a revenue collected for the special treatment.

All the cases of can be accommodated by using variable sized chunks. The presented system has used a variable sized chunk to cope with changes of popularity; chunks of different size could be used also within periods of static popularity to provide for differentiation as suggested in the aforementioned examples.

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.
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. 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, 1992.
13. 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.