Proxy Caching for Video on Demand Using Flexible Starting Point Selection

Wei Tu, *Student Member, IEEE,* Eckehard Steinbach, *Senior Member, IEEE,*
Muhammad Muhammad, and Xiaoling Li

***Abstract*—In this paper, we propose a novel proxy caching scheme for Video on Demand (VoD) services. Our approach is based on the observation that streaming video users searching for some specific content or scene pay most attention to the initial delay, while a small shift of the starting point is acceptable. We present results from subjective VoD tests that relate waiting time and starting point deviation to user satisfaction. Based on this relationship as well as the dynamically changing popularity of video segments, we propose an efficient segment-based caching algorithm, which maximizes the user satisfaction by trading off between the initial delay and the deviation of starting point. Our caching scheme supports interactive VCR functionalities and enables cache replacement with a much finer granularity compared to previously proposed segment-based approaches. Our experimental results show a significantly improved user satisfaction for our scheme compared to conventional caching schemes.**

***Index Terms*—Video on Demand, Video Streaming, Proxy Caching, Early Playout, Segment-based Caching, Popularity-aware Caching**

I. Introduction

Video on Demand (VoD) systems allow users to select and watch video over a network as part of an interactive entertainment system. Most of the VoD systems "stream" content, where video is consumed while being delivered, allowing viewing in real time. After users select a movie or television program, it is retrieved as quickly as possible and played on the client device. One of the main benefits of streaming video is that the users do not need to spend a long time and large storage cost to download the whole video file to the local disk.

Proxies are widely used in web browsing and can also be employed to improve the performance of video streaming. By deploying a proxy server close to the client, video playback with low latency and reduced network traffic can be achieved. In this work, we consider the scenario shown in Fig. 1, where a streaming proxy is located close to the clients in a Local Area Network (LAN). We assume that the connection between the clients and the proxy is characterized by high transmission rate and low latency. If the requested video content has already been accessed by one user and is cached on the proxy, the initial delay for the second and later users is significantly decreased compared to the case when content has to be loaded from the remote server. Since the proxy has only finite storage capacity, a dynamic cache management scheme has to be used to decide which videos or which parts of a video to cache.

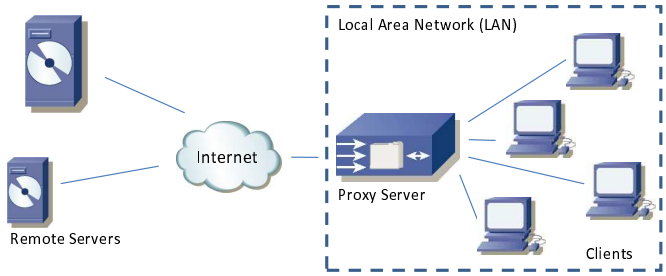


Fig. 1. Server-Proxy-Client network structure

A number of studies have been performed to explore the benefits of a proxy server for video streaming applications. Caching algorithms are proposed to minimize the data traffic between the server and the clients. For instance, *Video staging* approaches [1]–[3] cache the video portions on the proxy that exceed the given transmission rate so that a lower bandwidth is required to serve the video over a Constant Bit Rate (CBR) channel. In [4], both the characteristics of the video objects and the quality of the connections between the servers and the proxy are taken into account to determine the optimal video portions for caching. Miao and Ortega propose in their work [5] [6] caching strategies that consider an efficient control and usage of the client buffers. Buffer underflow and overflow are avoided with small initial delay. Both issues, the data traffic and the client buffer control, have been discussed and possible solutions are given by Oh and Song in [7] [8], where frames are selectively cached so that the normalized client buffer size is minimized. Furthermore, approaches for transcoding proxy caching are presented in [9]–[11], where the proxy caches different versions of the same video content to deal with heterogeneous user requirements.

Several partial caching approaches have been developed to decrease the initial delay experienced by a VoD user. Video content can be temporally divided into small units and some of these units are cached on the proxy to enable a fast playback.

Manuscript received Apr. 3, 2008; revised Dec. 5, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Pascal Frossard.

W. Tu and E. Steinbach are with the Institute of Communication Networks (LKN), Media Technology Group, Technische Universität München, 80290 Munich, Germany (e-mail: wei.tu@mytum.de; eckehard.steinbach@tum.de).

M. Muhammad is with German Aerospace Centre, Oberpfaffenhofen, Germany (e-mail:mmuhammed@hotmail.com).

X. Li is with Fujitsu Microelectronics Europe GmbH, 63225 Langen, Germany (e-mail: xiaoling.li@fme.fujitsu.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier

Prefix caching [12] [13] caches only the frames at the beginning of popular video clips to minimize the average initial delay. Segment-based approaches have been proposed to enable the cache update with a finer granularity. *Exponential Segmentation* [14] divides the video object such that succeeding segments are double in size and the replacement always starts from the largest segment. An extended version of it namely *Skyscraper Segmentation* has been later introduced in [15]. The *Lazy Segmentation* approach in [16] determines the segment length of a new incoming video as late as possible according to the user access record. The video is then divided into same length segments according to the mean viewing length and only the first several segments are cached when replacement is performed to free cache space. Segmentation of video objects can be in time domain but also in quality direction. For instance, in [17] and [18], caching algorithms for multilayer (scalable) video are proposed. A real implementation of the segment-based proxy caching infrastructure is reported by Chen *et al.* in [19] to show the feasibility of this approach. A combination of prefix and segment based approaches leads to the *chunk level cache replacement* methods, which further improve the flexibility of cache management. For instance, [20] combines neighboring units to form chunks. Each chunk consists of some prefix segments and some suffix segments. Suffix segments are never cached and the dropping of prefix segments starts from the tail of each prefix. The length of the chunks is predetermined according to the popularity of video objects, the higher the popularity, the larger the chunk size. A dynamic chunk size scheme is proposed in [21], where the chunk length in one video is fixed and is always an integer multiple of currently cached units. A larger chunk size is assigned to the videos with higher popularity and the replacement always starts from the end of the chunk having the smallest size. All above approaches assume that the playout always starts from the beginning of the video towards the end sequentially and random access to video content is not considered. This issue is addressed in a recent work by Wang and Yu [22], where a fragmental caching structure is proposed to enable interactive VCR functionality.

Popularity is one of the most important factors that should be considered during the design of an efficient caching strategy. Long-term movie popularity models in VoD systems have been derived in [23], which show how the popularity of video objects changes with time. In [24], real access log files show that for VoD services the popularity between video objects follows a Zipf Distribution [25], which says that most of the user requests are focused on a limited number of video objects. Different ways of calculating popularity are introduced and compared in [26]. *Static* counts the hitting frequency of a daily or weekly log file. *Accumulated* takes the history popularity into account and assigns different weights according to their time distance. *Dynamic* uses a sliding window to count the hitting rate in a short period of time. Popularity difference exists not only between video objects, but also inside one video. For instance, [27] reveals the internal popularity distribution of a video object by studying the log file of a commercial VoD system. Users start watching from the beginning of a movie sequentially and stop somewhere if the movie is uninteresting to them. In [28], the log files of a VoD system in the university also shows the diversity of popularity inside one video file. It reports that most of the users access the video randomly rather than using sequential playout. High access rates are always observed at the most attractive parts of the movie. Because of the random access, two popularity distributions should be considered, namely playing frequency and seeking frequency. As shown in [28], the two distributions are similar and the peaks are consistent.

Popularity-aware caching strategies, as described for instance in [4] [13] [26], use the popularity of served videos to decrease the initial delay and to improve the caching efficiency. Videos with high popularity are more likely to be cached and those with low popularity are not cached or only a small part is cached. Segment-based partial caching with explicit consideration of video popularity can significantly improve the performance of VoD systems as shown in [14]–[21]. However, most of the proxy caching algorithms assume that video files are always played from the beginning and continuously towards the end. Therefore, they put emphasis on the beginning of video files, while the rest receives less attention. Generally, this assumption does not always hold in practice. For instance, the log files of a real VoD system reported in [28] show that most of the video content in their system is randomly accessed instead of being played sequentially. Although [22] enables random access, only video level popularity is considered, which prevents a further improvement of caching efficiency.

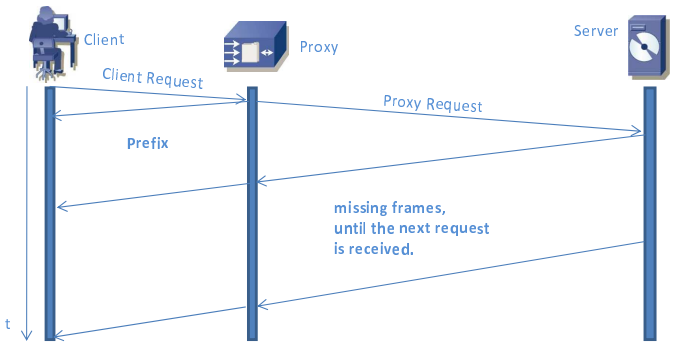


Fig. 2. Fast playback with prefix caching

In our previous work [29], a Popularity-Aware Partial cAching (PAPA) algorithm has been proposed. This approach is based on the results of an online subjective test, which has shown that VoD users prefer immediate feedback from the system even if the video does not start playing at exactly the desired starting point. Therefore, in PAPA, video files are divided into fixed-length segments (similar to the "chunks" in [20] and "fragments" in [22]), consisting of a prefix and a suffix. The prefix length in number of Groups of Pictures (GOPs) is determined by the network condition (round-trip-time, transmission rate, delay, jitter, etc.) between the proxy and the server where the requested video is stored and is chosen such that it is big enough to ensure continuous playback at the client. As every GOP is independently decodeable, the GOP size determines the granularity for random access. However, the playout in PAPA always starts from the beginning of a segment that a user request falls in. This leads to a small

deviation of the starting point (we call it "early start" in this paper), but as long as the prefix of the segment is completely available in the cache, the playback can start immediately. As shown in Fig. 2, when a request arrives at the proxy, the prefix of the segment where the requested GOP belongs to is forwarded to the user immediately if it is cached on the proxy. Meanwhile, a request is sent to the content server for missing video content. Downloaded frames are forwarded to the user when the whole prefix has already been sent. Furthermore, instead of evaluating the popularity of the whole video file, the popularity of every segment is evaluated, which leads to a more accurate popularity analysis and more flexible cache updates. When more cache space is required, PAPA drops all suffix frames before dropping any prefix frame to avoid waiting time for the client. Although the initial waiting time is minimized as long as all prefix frames are available, each segment has the same expected early start time without considering the large difference in their popularity. This limits the performance of PAPA.

In this work, we first introduce our improved subjective test environment and present more representative results, which further support the conclusion from our preliminary tests in [29] that users prefer a starting point deviation compared to initial delay. We give a mathematical expression for the relation between the initial delay and the early start time with respect to user satisfaction. Based on this observation, a Dynamic sEgment-based Caching Algorithm (DECA) is proposed. DECA inherits the concepts of "segment-prefix structure", "internal popularity of video" and "early start" from PAPA and further improves them. In DECA, instead of using a fixed chunk/segment size as in [22] and [29], the size of each segment is variable and changes as a function of the dynamically updated popularity. The individual segment size is determined by the estimated weighted user satisfaction, which is calculated from the popularity, the initial delay and the early start time of all GOPs belonging to this segment. Furthermore, DECA considers both the contribution of initial delay and early start, and makes a trade-off between them to achieve a highest averaged user satisfaction. To further improve the performance of a proxy caching system, the proxy cache is proposed to be divided into two parts, namely Level one (L1) cache and Level two (L2) cache. All newly incoming video fragments are temporally cached in the L1 cache following some traditional cache management schemes, e.g., LRU (Least Recently Used). The L2 cache is periodically managed with some segment-prefix based caching management schemes, e.g., DECA in this paper. Video fragments in the L1 cache that are frequently requested is then fetched to the L2 cache to replace those with decreasing popularity.

The remainder of this paper is organized as follows. In Section II, we present our subjective test platform as well as the obtained results. Based on the subjective test results, we describe the proposed DECA approach in detail in Section III. Simulation results are shown in Section IV and conclusions are drawn in Section V.

II. Subjective Test

Based on the results from our initial subjective tests in [29], we have drawn the conclusion that users prefer early start to initial delay. PAPA makes use of this observation and allows for early start to minimize the waiting time under the assumption that the segment length is fixed and small. However, using short segment lengths also means that more content needs to be cached, which decreases the caching efficiency. In this section, we present new subjective tests that allow us to determine the expected user satisfaction as a function of initial delay and early start time. Equipped with this functional relationship, the proxy is able to serve a user request with the mode which leads to higher satisfaction. Furthermore, in the initial tests we have shown the test videos to the subjects explicitly with the expected starting point and the real starting point, which is somewhat unfair to the early start mode. In the new test, we improve the interactivity with the system by adding the functionality of random access using a slide bar.

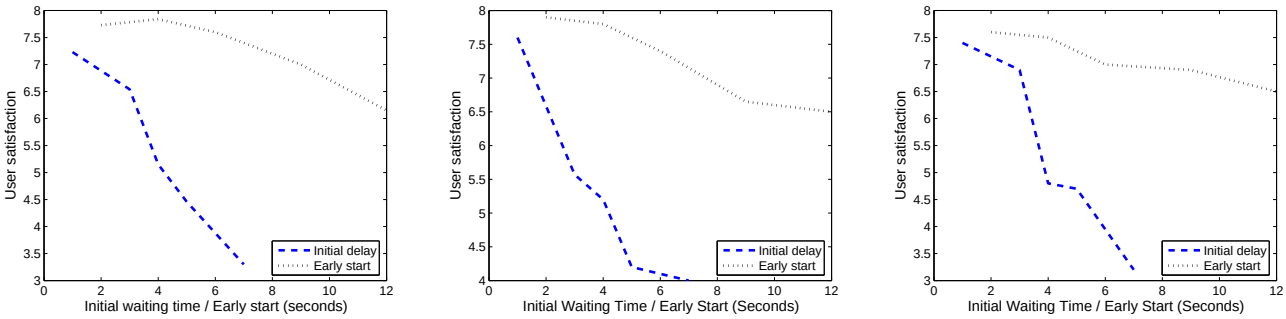
A. Test Setup

To develop a media player with full functionality and friendly user interface, the open source Video LAN/VLC .Net bindings [30] is used as a core player and C# is used to develop the interface. Our media player, shown in Fig. 3, is called "TUMPlayer", which has the functionalities of *Play*, *Pause*, *Stop*, *Fast Forward/Rewind*, and controllers for *Volume*, *Thumbnail View* and *Full Screen*.

Fig. 3. User interface of the TUMplayer

Three videos with different characteristics have been included in the tests: *News*, *Sport* and *Movie*. The *News* is freshly captured from CNN TV news. The football match in the *Sport* category is the final of "Copa Libertadores 2007". Finally, the film "Shrek-I" is used to represent the *Movie* category. All three video clips are encoded using MPEG-1 with CIF@25fps. More properties of the test videos are shown in Table I.

The subjective test is conducted as follows. When a volunteer starts the test, two frames selected from two popular scenes are popped up on the left side of the player as shown in Fig. 3. The test person is told to access the popular scene



(a) News (b) Sport (c) Movie

Fig. 4. Results of the subjective VoD performance evaluation test

TABLE I
PROPERTIES OF TEST VIDEOS

Video Name	Total number of frames	Duration
News	39,504	00:26:20
Sport	72,441	00:48:17
Movie	131,976	01:27:59

by clicking on the pop-up thumbnail frames. After clicking, according to our specification, the system serves randomly with one of the following modes:

1) The playout of the video starts after an initial delay of 1, 3, 4, 5 or 7 seconds (modes 1 to 5) exactly at the selected scene. In addition, a "Buffering..." text message is displayed in a panel at the bottom of the player.
2) The playout starts immediately without any initial delay but with a shift of 2, 4, 6, 9 or 12 seconds prior to the selected frame (modes 6 to 10).

The test person is asked to wait until the scene he/she clicked on appears in order to make sure that the early start becomes noticeable. After watching the given scenes, the test person is also asked to access the video clip randomly using the slide bar at the bottom of the player to get an additional impression about the interactivity of the system. Finally, a score is given by the test person for one video sequence under one operation mode, which represents his/her satisfaction with the system.

B. Results

Fig. 4 shows the obtained results of our subjective tests. The X-axis represents the initial waiting time or the early start time in seconds. The Y-axis shows the user satisfaction on a scale from 1 to 10, where 1 indicates the worst user experience and 10 is the best. All points on the curves are derived by averaging the scores from 28 test persons. The dashed curve represents the client satisfaction as a function of the initial delay and shows that the satisfaction of the user declines rapidly with increasing waiting time. The dotted curve illustrates the user satisfaction as a function of the early start time and shows a much slower degradation of user satisfaction. Hence, we can draw the conclusion that the users are more comfortable with the early start than the initial delay. In other words, when clients are searching for some particular content, they pay more attention to the initial delay and a small shift of the starting point is more tolerable. Please note that the duration of the movie and the slide bar navigation play an important role in the tests. In our tests, following the design of real VoD systems [28], long videos are employed. As the size of the slide bar in the player is fixed, the shift of the slider for the same playout duration varies according to the length of the video. A very small movement is observed in our case, which makes the early start unnoticeable during slide bar interaction. On the contrary, clips with very short duration will show a large movement of the slider on the track bar, which is then annoying as early start becomes observable when searching with the slide bar. In this case, we can offer coarser access granularity on the segment level by disabling the requests to later GOPs in the segment.

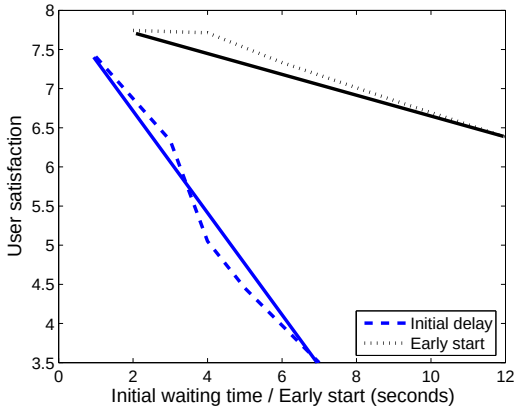


Fig. 5. Averaged user score and approximated user satisfaction model

Another observation we can make in Fig. 4 is that different types of videos lead to similar results. This encourages us to derive a universal user satisfaction model that is independent of the video type. We average the results for the *News*, *Sport* and *Movie* videos and obtain the new user satisfaction curves shown in Fig. 5. We approximate the curves in the following with two linear functions:

$$G_{WT} = max(0, -0.653 \cdot t_{WT} + 8) \quad (1)$$

$$G_{ES} = max(0, -0.137 \cdot t_{ES} + 8) \quad (2)$$

where t_{WT} and t_{ES} represent the initial delay and the early start time in seconds, respectively. G_{WT} and G_{ES} denote the

user satisfaction score for the two different serving modes, respectively. For zero waiting time or zero early start time, a score of 8 is obtained in (1) or (2). This reflects that during the subjective test, the test persons were too conservative to give the full points. Therefore, in the following, we assume 8 to be the highest score.

III. Caching Algorithm with Dynamic Segment Structure

In this section, we first present the variable segment structure and some important definitions for the Dynamic sEgment-based Caching Algorithm (DECA), followed with a detailed description of the algorithm. A two-level caching framework is then introduced, where the proposed DECA can be employed to achieve even higher user satisfaction.

A. Segment-Prefix Structure

As shown in [22], more cache resources should be assigned to the video fragments with high popularity. Therefore, instead of using a fixed-segment structure as in [29], a more flexible segment-prefix structure is proposed, so that video fragments with higher popularity can have smaller segment size to enable small or even no initial delay and early start. On the other hand, larger segments are mostly used for those video parts with low popularity. This is different to the proposal in [20], where the popular videos have larger chunk size. Furthermore, the smallest replacement unit is defined as one GOP in our work rather than "any multiple of disk block size" in [20].

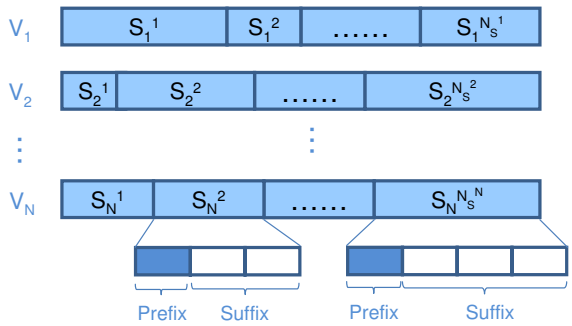


Fig. 6. Variable size segment structure

Fig. 6 shows the segment structure with variable segment size. As illustrated, there are N videos altogether, namely V_1 to V_N, each of them has N_S^v $(v = 1 \dots N)$ segments. Generally speaking, different videos may have different segment size, and the segment size within the same video may also vary significantly. Each segment consists of two parts: a prefix and a suffix. The length of the prefix or suffix is expressed as the number of GOPs it contains. Take the N-th video in Fig. 6 as an example. The second segment of V_N consists of 1 prefix GOP and 2 suffix GOPs, while the last segment comprises 1 prefix GOP and 3 suffix GOPs. The length of the prefix (L_P) is determined by the network condition between the server and the proxy. For a certain video file, we assume that it will be retrieved from only one content server and hence, the same network condition applies to all segments of a video. As a result, they have identical prefix length. For simplicity but without loss of generality, we assume that all videos have the same prefix length in this work. The minimum segment size is equal to the prefix length, which is also the finest achievable random seek granularity for an immediate playout unless we have L_P successive GOPs from the requested point cached on the proxy.

The variable size segment structure empowers the replacement algorithm to keep up with the changing popularity distribution and also provides a way to adjust the granularity of random access. Video fragments with high popularity have smaller segment size, therefore finer granularity and vice versa. With this flexible structure, we can also ensure that all prefix GOPs are cached by adjusting the size of individual segments. Therefore, if the early start mode is selected by the system, no additional waiting time will be experienced by the user. Otherwise, if the initial delay mode is more favorable, the playback will start exactly at the requested point.

B. Serving Mode Selection

As mentioned in the last section, different modes (i.e., initial delay or early start) can be selected to serve the user requests. In this section, we introduce how the waiting time and early start time can be calculated for the user request to a particular GOP. Based on that, we determine the optimal serving mode.

The expected waiting time for a particular request is the time needed to download L_P successive GOPs from the requested starting point if any of them are not cached on the proxy. Only when they are all available, the playout can be started. These L_P GOPs might go across the segment boundary and include some prefix GOPs of the next segment, which should always be cached according to the design of the prefix-segment structure. Therefore, only the loading time for missing suffix GOPs has to be considered. We obtain the waiting time for a request to the i-th GOP of segment j in video v as

$$t_{WT}^{(v,j,i)} = \sum_{n=max(L_P+1,i)}^{min(L_S^{(v,j)},i+L_P-1)} \frac{S_G^{(v,j,n)}}{R^v}, \tag{3}$$

where $S_G^{(v,j,n)}$ is the size of the n-th GOP in the j-th segment of video v. $L_S^{(v,j)}$ is the length of segment j in number of GOPs and L_P is the length of prefix in number of GOPs. R^v is the transmission rate between the proxy and the server where video v is stored.

The early start time is determined by the distance between the requested GOP and the first GOP in the segment. It can be calculated as

$$t_{ES}^{(v,j,i)} = \frac{L_G}{r_f} \cdot (i-1), \tag{4}$$

where L_G is the length of one GOP in number of frames and r_f is the frame rate of the video sequence. The ratio L_G/r_f converts the early start time in number of GOPs to the early start time in seconds.

In DECA, the playout does not necessarily start from the beginning of a segment. For large segments, if the desired starting point is too far away from the beginning of the

segment, the waiting mode might be selected as the serving mode. This is because in this case, early start will lead to a low user satisfaction, which might be even lower than the user satisfaction obtained for downloading of the requested part from the remote server. Let's assume that the early start time would be 12 seconds and alternatively the waiting time would be 1 second. In this example(compare Fig. 5), users would prefer waiting for 1 second rather than encountering a starting point deviation of 12 seconds. On the other side, when the requested GOP is not far away from the beginning of the segment it belongs to, the early start mode is typically preferred. Once the early start time and the waiting time for the requested GOP is obtained from (3) and (4), we can calculate the scores for the two modes by evaluating the score functions (1) and (2). The two scores are compared and the mode leading to higher user satisfaction is finally selected, which is presented as

$$G^{(v,j,i)} = max(G_{ES}^{(v,j,i)}, G_{WT}^{(v,j,i)}). \tag{5}$$

C. GOP Level Popularity

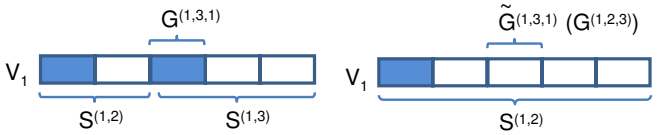
As a lot of video content is randomly accessed by VoD users [28], different parts of a video may have very different popularity [27]. For example, fragments with goals in a football match are usually visited more frequently than other fragments. Suppose we have two videos V_1 and V_2 cached on the proxy. Let's further assume that the overall popularity of V_1 is higher than the popularity of V_2. However, the most popular fragment of V_2 has a popularity exceeding most parts of V_1. If a cache replacement algorithm works with video level popularity, video frames in V_2 will always be removed before V_1 because it has a lower overall popularity, even the most popular part of it might not survive in this process. Obviously, a more efficient algorithm should delete those parts in V_1 with a lower popularity rather than the most popular part in V_2. As GOPs are independently decodeable, they define the finest access granularity. Therefore, instead of evaluating the popularity for the entire video as in [4] [26] [13], we measure the popularity for every GOP during a predefined time interval.

D. Cost Calculation

Before we explain our proposed cache management scheme, we introduce an important metric called "cost", which is used during the subsequent merging process to build up the variable segment structure shown in Fig. 6. The "cost" mentioned above represents the price we have to pay for one byte of free cache space if two neighboring segments are merged together, of course the cheaper the better. The pairwise cost is a function of three different parameters. Popularity is the first parameter. We want to keep more popular video fragments, hence, the higher the popularity, the bigger the cost should be. The second aspect to consider is the user satisfaction. When two segments are merged, user satisfaction degrades because of the deletion of the second segment and therefore larger early start or more waiting time is experienced. We hope this degradation to be as small as possible, therefore, the bigger the decrease of the satisfaction score is, the larger the cost should be. The last issue is the released cache space. The larger the free space created after a merger, the smaller the cost should be. By the merger of two neighboring segments, the waiting time of the last L_P GOPs in the first segment increases because the prefix of the second segment is now missing. Meanwhile, both the waiting time and the early start time for the GOPs in the second segment might also differ. Therefore, the user satisfaction degradation of GOPs in both segments should be considered. Based on the above arguments, we define the cost for segment pair j of video v as:

$$C^{(v,j)} = \frac{\sum_{s=j}^{j+1} \sum_{i=1}^{L_S^{(v,s)}} (p^{(v,s,i)} \cdot \Delta G^{(v,s,i)})}{\Delta B^{(v,j)}}, \tag{6}$$

where $p^{(v,s,i)}$ is the popularity of GOP i in segment s. $\Delta G^{(v,s,i)}$ denotes the decrease of the user satisfaction score for the i-th GOP in segment s if we merge the pair together. The summation of this weighted user satisfaction degradation gives the total price of this merger. $\Delta B^{(v,j)}$ is the corresponding increase in free cache space. A large cost value means the pair may have a high popularity or the user satisfaction declines severely or only limited free cache space is generated after the merger. We tend to keep such a pair on the proxy.



(a) Before merging (b) After merging

Fig. 7. Example of cost evaluation for segment merging

The increase in free space ΔB is easy to obtain. It is simply the size of the prefix of segment $S^{(1,3)}$ of video V_1 in the example shown in Fig. 7(a). To get the decrease in user satisfaction ΔG of GOP i in $S^{(1,3)}$, we need to calculate the user satisfaction score of GOP i before and after the merger, which is $G^{(1,3,i)}$ and $\tilde{G}^{(1,3,i)}$ in Fig. 7, respectively. Please note, $\tilde{G}^{(1,3,i)}$, when i=1, can also be presented as $G^{(1,2,3)}$ in Fig. 7(b), the third GOP of $S^{(1,2)}$ after merger. It can be obtained by using (3)(4)(5) and considering the merged segment $S^{(1,2)}$ in Fig. 7(b). Finally, the decrease of user satisfaction for GOP i by this merger can be calculated by:

$$\Delta G^{(1,3,i)} = G^{(1,3,i)} - \tilde{G}^{(1,3,i)}. \tag{7}$$

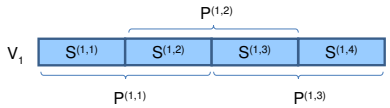
Suppose we would like to calculate the user satisfaction degradation of the first GOP in $S^{(1,3)}$. Before the merger, it is the first GOP of the segment and belongs to the prefix. Therefore, it can be played without delay and starting point deviation, which achieves a user satisfaction score of 8. After the merger, it becomes the third GOP in segment $S^{(1,2)}$ in Fig. 7(b). When it is requested, the proxy can either download the missing part from the remote server and then deliver it to the user, or send the prefix to the user instead and meanwhile download the missing GOPs. The former option results in 1 second initial delay on average while the latter option leads to an early start of 2 seconds but no initial delay. We assume in this example that the frame rate in Hz equals to the GOP

length in number of frames and the transmission rate between the server and the proxy equals to the mean rate of the video stream. According to (1) and (2), the user satisfaction score for 1 second waiting time is 7.347 and the score for 2 seconds early start is 7.726. Hence, the early start mode will be chosen to serve the request to this GOP, which results in a score of 7.726. The score degradation $\Delta G^{(1,3,1)}$ can then be calculated with (7) and it equals to 0.274. The score degradation for the rest GOPs can be obtained in the same way.

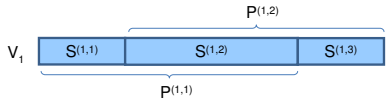
E. Cache Replacement Algorithm DECA

In this section, we show how the variable size of each segment is determined by recursively merging of neighboring segments. After the algorithm is employed to create the variable segment structure, the proxy cache is updated accordingly by removing all suffix GOPs and prefetching all prefix GOPs. The updated cached content achieves an improved user satisfaction score according to the long term accumulated popularity. Therefore, we call this procedure "cache update" in the following.

The cache update algorithm is invoked for regular maintenance of the proxy cache. With the current segment structure and the popularity distribution available to it, the replacement algorithm works as follows. It first browses through all videos, checks the neighboring segments in a video pairwise, calculates the merging cost in (6) for each pair and saves it to an array. After all the segment pairs have been checked, the algorithm sorts all cost values in ascending order and starts merging from the first pair, i.e., the one with the smallest cost. Merging a segment pair means to join the two segments together and make one bigger segment out of them. All GOPs of the second segment in this pair become suffix GOPs and are completely removed to release cache space.



(a) Before merging



(b) After merging

Fig. 8. Example of pair information update for segment merging

Every time the DECA algorithm is called, it runs iteratively and the pair with the lowest cost is merged in each loop. As the two segments in this pair are now replaced by a new segment, the cost information of the pairs with neighboring segments has to be updated. For example, as shown in Fig. 8(a), when pair $P^{(1,2)}$ is selected and merged, a new segment (i.e., $S^{(1,2)}$ in Fig. 8(b)) is created. The pairs $P^{(1,2)}$ and $P^{(1,3)}$ before the merger consist of segments $S^{(1,2)}$ and $S^{(1,3)}$, which no longer exist afterwards. Therefore, their cost has to be recalculated. Please note, pair $P^{(1,3)}$ in Fig. 8(a) becomes the second pair $P^{(1,2)}$ in Fig. 8(b). After updating the information of related pairs in the cost array, the next loop starts. This procedure continues until all remaining content can be stored in the cache.

When DECA is run for the first time, an initial segment structure is built up, where all the GOPs are intact and every L_P of them are grouped together to form one initial segment. The merging operation starts from such an initial segment structure and stops when a target cache size is reached. As the merging operation goes on, there will be fewer segments in the cache. The final segment structure resulting from the merging operation comprises segments of different lengths. They all have the same prefix length whereas their suffix length differs. After running DECA using the initial segment structure, all suffix parts currently cached will be deleted. If the new prefix GOPs are not cached, they have to be reloaded from the remote server.

Generally speaking, the cache replacement algorithm can use the output of the last round and perform further combinations or updates based on that. If the popularity distribution between two successive updates stays similar, none or only partial update of the cache is needed. In this case, we keep most of the segment pairs unchanged and only work on the fragments which have large changes in their popularity. For those having a decreased popularity, segment pairs will be merged to leave some space. Meanwhile, segments with increasing popularity can also be splitted into two small segments (e.g., evenly), recursively. In our experiments, for the sake of simplicity, the replacement algorithm always takes an initial segment structure as its input. However, we found from our experiments that these two (the normal and the simplified) approaches lead to similar results.

F. Performance Evaluation Metric

In this section, we introduce the metric that will be used to evaluate the performance of DECA. We first calculate the mean early start time and the mean waiting time of the system. Based on that, we obtain the final score according to (1) and (2), which reflects the user satisfaction for the VoD system.

In Section III-B, we have shown how the optimal mode can be determined for a particular user request. As for every request, only one mode is selected, either the waiting time or the early start time is zero. Hence, we define a threshold T^v, which represents the switching point between the two modes for video v. When the requested GOP has an index inside a segment larger than T^v, the waiting mode leads to higher user satisfaction and the early start time is zero. Otherwise, the early start mode is preferred and the waiting time becomes zero. It can be obtained from (1) and (2) as

$$T^v = 4.77 \cdot \frac{L_P^v \cdot S_G^v \cdot r_f}{L_G \cdot R^v}, \tag{8}$$

where S_G^v is the average GOP size of video v, which is used here as an approximation. We then rewrite (3) and (4) as

$$t_{WT}^{(v,j,i)} = \begin{cases} 0 & \text{if } \ i \leq T^v \\ \sum_{n=max(L_P+1,i)}^{min(L_S^{(v,j)},i+L_P-1)} \frac{S_G^{(v,j,n)}}{R^v} & \text{if } \ i > T^v \end{cases} \tag{9}$$

and

$$t_{ES}^{(v,j,i)} = \begin{cases} \frac{L_G}{r_f} \cdot (i-1) & \text{if } i \leq T^v \\ 0 & \text{if } i > T^v \end{cases} \quad (10)$$

Based on (1) and (9), the mean waiting time of the whole system can be determined by

$$E_{WT} = \sum_{v=1}^{N_V} \sum_{j=1}^{N_S^v} \sum_{i=1}^{L_S^{(v,j)}} t_{WT}^{(v,j,i)} \cdot p^{(v,j,i)}, \quad (11)$$

where $p^{(v,j,i)}$ represents the normalized access frequency of GOP i in the j-th segment of video v. Similarly, based on (2) and (10) the averaged early start time of the whole system is

$$E_{ES} = \sum_{v=1}^{N_V} \sum_{j=1}^{N_S^v} \sum_{i=1}^{L_S^{(v,j)}} t_{ES}^{(v,j,i)} \cdot p^{(v,j,i)}. \quad (12)$$

The mean waiting time and the mean early start time are important performance parameters of the system. However, with any one of the two, we can not give a comprehensive evaluation of the system. Instead, both aspects have to be taken into account simultaneously. Therefore, the user satisfaction score in (5) is adopted and the overall performance of the VoD system is calculated as

$$G = \sum_{v=1}^{N_V} \sum_{j=1}^{N_S^v} \sum_{i=1}^{L_S^{(v,j)}} G^{(v,j,i)} \cdot p^{(v,j,i)}. \quad (13)$$

G. Two Level Caching Structure

In many conventional VoD cache management schemes, the whole cache is managed by a single algorithm and hence might lead to unnecessary or improper replacement. To improve the efficiency of cache management, a two-level cache structure can be employed. As shown in Fig. 9, the entire cache is divided into two parts, namely L1 cache and L2 cache, which can properly manage the caching of videos with short-term popularity and long-term popularity. The L1 cache is used to cache those videos which have a very high request frequency in a very short time period. The L2 cache is reserved for video content that has high popularity over a longer time and the proposed DECA approach is a good candidate to manage it.

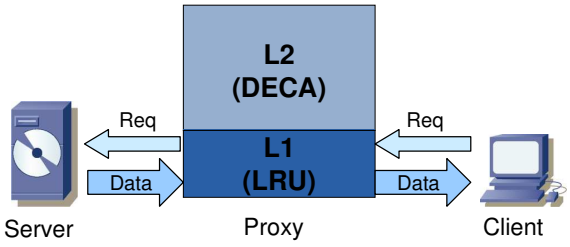


Fig. 9. Two-level cache structure

When a user request arrives at the proxy, the proxy checks the L1 cache and L2 cache as a whole. If there is a cache hit, the proxy starts forwarding the requested content immediately to the client. If the result is a cache miss, the proxy has two options. It can either wait for the loading of the missing part from the server or start from the closest cached prefix. For the first option, the missing GOPs are requested from the remote server and delivered to the client after some delay. For the second option, the closest prefix that is completely available before the intended starting point is sent to the client right away. In both cases, missing frames in the requested segment and successive segments are retrieved from the remote server to achieve a continuous playback. All received video content passes through and is cached in the L1 cache before being delivered to the client. As only missing video content needs to be downloaded, it will not be duplicated in the L1 cache and the L2 cache. Once the L1 cache is full, classic caching update algorithms (e.g., LRU) can be used to release some space for newly downloaded content. The L2 cache is regularly managed using DECA described in the previous sections. Unpopular fragments in the L2 cache are either exchanged with popular ones in the L1 cache or replaced by popular video content fetched from the content server. The update period of the L2 cache is normally much longer than that of the L1 cache, and could for instance be daily or weekly. When the distributions of popularity at two successive update time points are very similar, the update of L2 cache can be even skipped.

With the two level caching structure, the proxy cache can be managed in a more efficient way. For instance, when a video fragment is requested for the first time, it will be fully loaded from the remote server and temporally cached in the L1 cache. If it is popular, the frequent user requests will prevent it from being replaced, although the accumulated hitting rate of this fragment is still low. As most of the frequently requested video content between two updating points is either cached in the L2 cache as prefixes or temporally cached in the L1 cache when the L2 cache is updated using DECA, little additional traffic between the proxy and server will be generated. On the contrary, if the requested video fragment is of low popularity and is just occasionally visited, it will be replaced soon after being temporally cached in the L1 cache and it has no chance to enter the L2 cache. This ensures that the cached content in the L2 cache will not be replaced by those fragments which are rarely requested.

When the two-level cache structure is employed, both the mode selection and evaluation metrics should also be modified. If the suffix GOPs of a segment have been accessed recently and are cached in the L1 cache on the proxy, the waiting time for loading these already cached GOPs should be subtracted, which leads to a shorter initial waiting time.

IV. Simulation Results

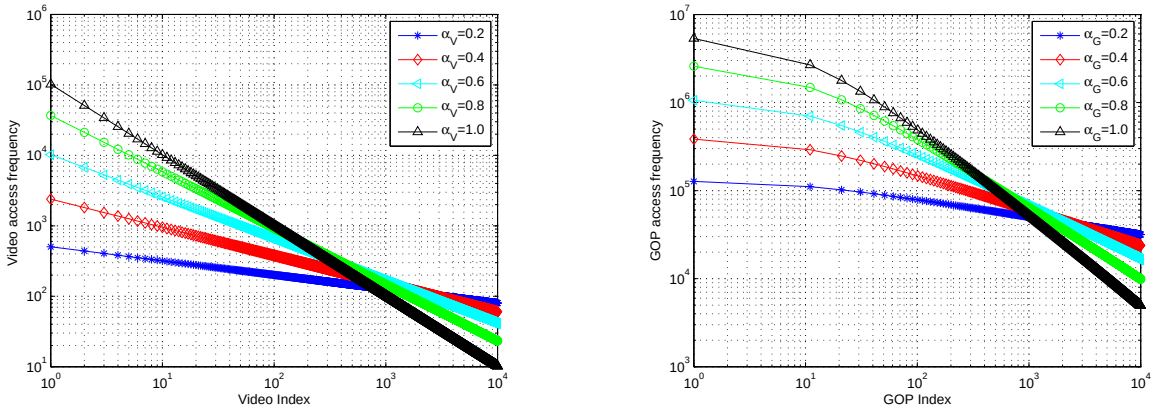
In this section, we first evaluate the performance of our proposed DECA approach for different parameter setups. Then, we compare the performance of DECA to comparison schemes.

A. Simulation Setup

In this work, we use online video trace files [31] instead of real video sequences. Necessary information about all the

TABLE II
PROPERTIES OF THE VIDEOS USED IN THE SIMULATION

Video Name	GOP length	#B frames	QP	Resolution	Frame rate	Length(min)	Size(MB)	Bitrate(kbps)
Silence of the Lambs	16	3	28	CIF	30	30	159	144
Star Wars IV	16	3	28	CIF	30	30	161	156
NBC 12 News	16	3	28	CIF	30	30	612	439
Tokyo Olympics	16	3	28	CIF	30	74	902	306



(a) Zipf distribution

(b) K-Transformed Zipf distribution

Fig. 10. Example of video and GOP level access frequency, which are used as popularity distributions in our experiments

videos hosted on the remote server is included in the trace files. We use four sequences in our simulation, however, a larger number of test sequences can be easily added and they achieve similar results. The videos used in the simulation are encoded with the H.264/AVC codec and Table II shows the major properties of them.

The second issue is how to obtain the request events to represent the user behavior. We assume that the user request frequency for video objects follows a Zipf distribution [25], and the user request frequency for object i is calculated as

$$f_i = \frac{1}{i^{\alpha} \cdot \sum_{j=1}^{m} j^{-\alpha}}, \tag{14}$$

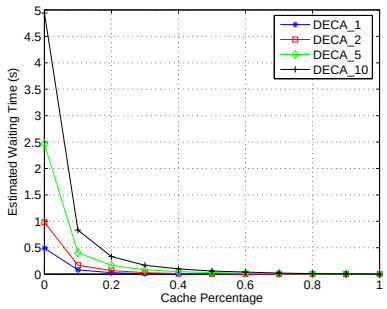
where m is the number of distinct objects. α is the parameter to adjust the skewness of the Zipf distribution, which is larger than 0 and less than or equal to 1.0. The larger the α, the bigger the difference of popularity between objects. According to [27], the internal popularity of a video (i.e., the access frequencies of all GOPs) follows a K-Transformed Zipf distribution. Therefore, a Zipf RNG (random number generator) and corresponding transformations with two positive constants $K_x = 10$ and $K_y = 400$ are employed to generate the requests to the GOPs in all videos. We assume that the GOP in the middle of a video object has the highest access frequency and it decreases monotonically towards the beginning and the end of the video object. We mark the α for the video level Zipf-distribution as α_V and for the GOP level K-Transformed Zipf distribution as α_G. The access frequency with different α values for both distributions is shown in Fig. 10, taking 10000 units as an example. In our experiments, without specification, both α are set to be 0.8 [22]. Please note that real user request log files can also be adopted into our algorithms and should achieve similar results.

B. Performance of DECA

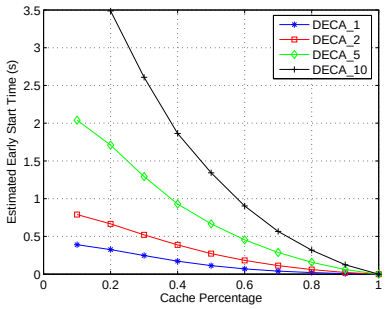
In this section, we investigate the performance of the DECA approach for different scenarios. In Fig. 11, we present different performance metrics as a function of the percentage of total video content that can be cached on the proxy. *DECA_M* in the figure represents the DECA algorithm with a prefix length of M GOPs.

Fig. 11(a) shows the mean waiting time for different prefix lengths, which corresponds to different network conditions between the proxy and the server(s) as mentioned in Section III-A. When the cache percentage is 0%, i.e., no video content is cached, any requested video content needs to be loaded from the remote server. This leads to a larger initial delay for *DECA_10* because it has more data as prefix that needs to be loaded from the remote server compared to DECA with smaller prefix size. Thanks to the dynamic segment structure of DECA, all prefix GOPs can be cached, which leads to a significant decrease of waiting time when the cache percentage increases. When the caching rate is very small, e.g., smaller than 10%, although the prefixes are all available, the segments are too long so that "wait for loading" is more preferable than a large deviation of starting point. Therefore, some initial waiting time is observed for the requests to some middle- to unpopular parts. However, zero initial delay can be achieved as soon as about 60% of the video content is cached.

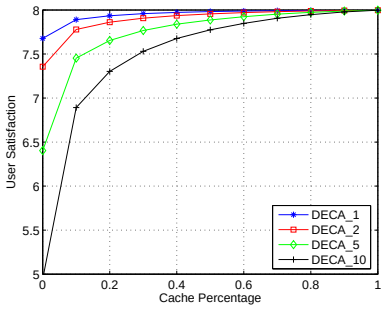
Fig. 11(b) illustrates the expected early start time as a function of the caching percentage when DECA is employed to manage the cache. The early start time decreases when the cache percentage increases, as caching more content on the proxy results in fewer segment mergers and thus smaller segment size on average. Please note, a cache percentage of 0% is a special case, where nothing is cached and the early



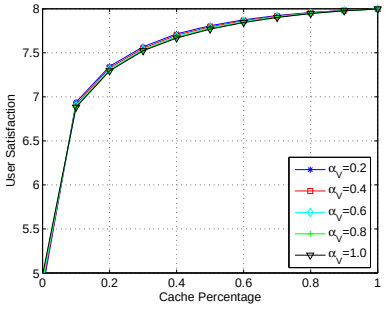
(a) Waiting Time (α_V=0.8, α_G=0.8)



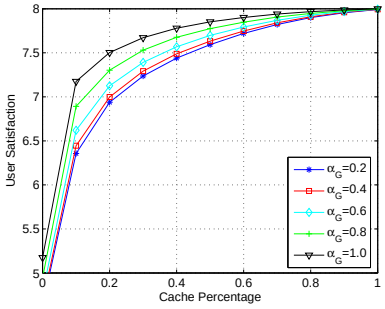
(b) Early Start (α_V=0.8, α_G=0.8)



(c) Final Score (α_V=0.8, α_G=0.8)



(d) Final Score (L_P=10, α_G=0.8)



(e) Final Score (L_P=10, α_V=0.8)

Fig. 11. Performance of DECA as a function of cache percentage

start is disabled. DECA with smaller prefix size needs less cache size for each segment and therefore has more segments at the same caching percentage. This leads to an overall smaller segment size throughout the video and thus smaller expected early start time.

We show the final score of DECA for different prefix lengths in Fig. 11(c). The final score of DECA can be obtained using (13), where the optimal serving mode has been determined by comparing the achievable scores of waiting time and early start time. *DECA_1* has the smallest waiting time and the shortest early start time, therefore, achieves the highest final score at all caching percentages. When more content is cached (i.e., larger than 50%), always the same mode will be selected, which leads to a very close performance between all the curves in Fig. 11(c). When the caching percentage is zero, i.e., all requested content needs to be downloaded from the server, *DECA_1* still performs the best because of its smallest prefix size and therefore least amount of data to cache before the playback starts.

Fig. 11(d) and Fig. 11(e) illustrate the user satisfaction score for the (K-Transformed) Zipf distribution with different α values when the prefix length L_P is equal to 10 GOPs. The user satisfaction shows only a small change when α_V varies, because the limited number of videos involved in the simulation and thus the change of popularity among videos is not dramatic. When we fix α_V, a lower user satisfaction score is clearly observed when α_G decreases. This is because the difference of popularity between video GOPs is not so dramatic for small α_G values. Therefore, caching the most popular video parts contributes less to the overall performance than when $\alpha_G = 1.0$. Nevertheless, DECA shows a consistent performance independent of the selection of α for the assumed request distribution. In the following simulations, when we investigate the influence of α to the final score of all schemes, we will set α_V to be 0.8 and consider only α_G.

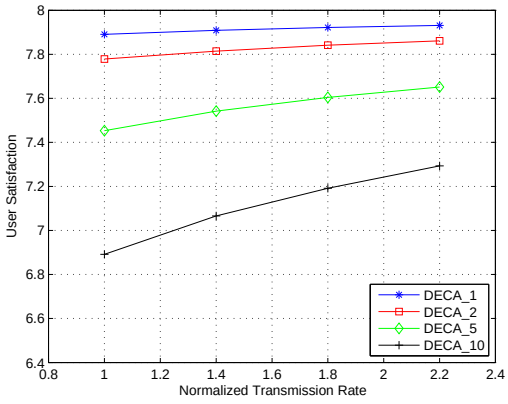


Fig. 12. User satisfaction as a function of the available transmission rate at a cache percentage of 10%. (α_V=0.8, α_G=0.8)

In the above simulations, we have assumed that the transmission rate between the proxy and content server is equal to the mean rate of the video stream. However, if the available transmission rate is larger than the mean rate, the waiting time to load the same amount of data decreases. As the early start time has no relation with the transmission rate, the "initial waiting" mode becomes more attractive and will be selected more frequently. With a faster connection to the server, the overall performance of the system improves.

Fig. 12 illustrates the achievable user satisfaction score

of *DECA_M* as a function of the available transmission rate between the server and the proxy. The X-axis shows the normalized transmission rate, which is the ratio of the available transmission rate over the mean rate of the bitstream. Naturally, the larger the transmission rate is, the higher the achievable user satisfaction. As can be observed from Fig. 12, the increase of transmission rate is most helpful to the segment structures with a larger prefix size, where *DECA_10* shows the biggest improvement. This is because a larger absolute time saving for loading the missing GOPs is achieved for *DECA_10*, which results in a remarkable improvement of user satisfaction according to (1). Because of the limited space, we only show the results when α_G equals 0.8. When smaller α_G values are used, similar results are observed, however, the improvement by increasing the rate is even larger. This is because in DECA, segments with high popularity are of very short length (i.e., the segment only consists of a prefix) and requests to them can be served with neither initial delay nor early start. Therefore, when α_G becomes smaller, more requests benefit from the higher transmission rate resulting in smaller waiting time, which leads to an improvement of the overall user satisfaction.

According to Section III-G, the whole cache is divided into two levels. DECA is employed to manage the L2 cache, while the L1 cache uses the LRU algorithm for the updating. We have shown in the above the performance of DECA, when it controls the whole cache for different simulation setups. The influence of the L1 cache is not shown here because no real access log files are available to us and the Zipf distribution can only give us a statistic hitting rate without the information of request order. However, the performance improvement when including L1 cache is obvious when the popularity changes significantly. For example, if there is a new video published by the server after the last run of DECA, by assigning 20% of the cache as L1 cache, this popular video might be fully cached on the proxy. It leads to some degradation from the DECA perspective because of the decreased caching size. However, a significant gain can be obtained by employing the L1 cache, which leads to high user satisfaction for this popular video. On the contrary, if the popularity change is very small and no new video comes in during the two updating points, letting DECA work for the whole cache leads to better results.

C. Performance of PAPA

In this section, we show the performance of the PAPA approach in [29], on which DECA builds. As mentioned in the introduction, PAPA is a cache management scheme that uses a fixed segment-prefix structure. We use PAPA as one of the comparison schemes in the next section.

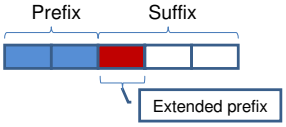


Fig. 13. One complete suffix GOP is cached

Here, we implement an improved version of PAPA compared to the original description in [29]. Instead of always starting the playout from the beginning of one segment, a closer starting point can be considered if some suffix frames are also cached to form an extended prefix as shown in Fig. 13. In this case, the playout can be started from a GOP which has a distance of L_P GOPs to the first unavailable suffix. By serving the client from a closer point to the desired starting point, the expected early start time is decreased. In the extreme case, if all video content is cached on the proxy, no early start will be experienced. For example, to respond to a user request to the fourth GOP in Fig. 13, instead of starting with the first GOP in this segment, the playout can start with the second GOP, which together with the third GOP act as the prefix for this request. This leads to again no waiting time, but smaller shift from the desired starting point.

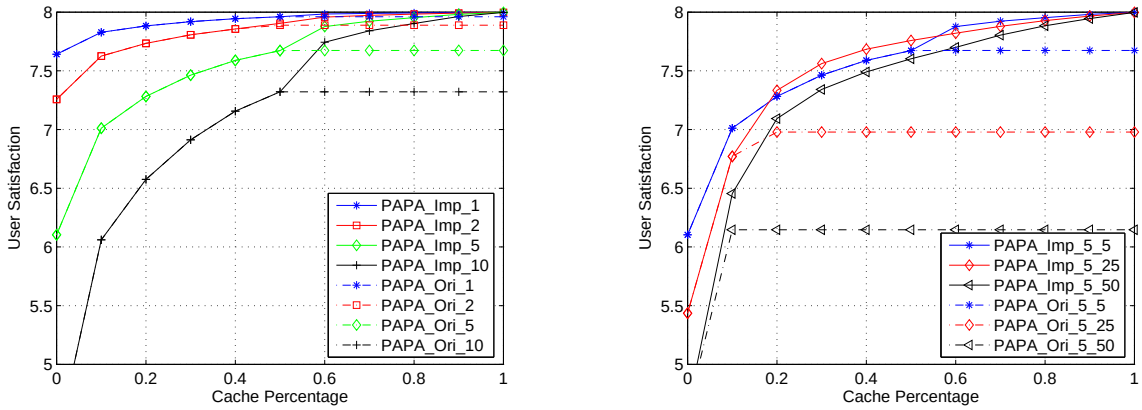
Fig. 14 shows the performance of PAPA as a function of the percentage of cached content. The dash-dotted curves and the solid curves in the figure represent the original PAPA approach (*PAPA_Ori*) proposed in [29] and the improved version of PAPA (*PAPA_Imp*) introduced above, respectively. In Fig. 14(a), the score increases for the improved PAPA approach when the cache percentage exceeds 50% because we set the suffix length equal to the prefix length in this experiment. When more than 50% of the video content can be cached, all prefix GOPs are cached and some of the suffix GOPs are also cached, which enables a closer starting point. This decreased early start time achieved by the improved PAPA approach leads to a significant improvement of user experience. Fig. 14(b) shows the influence of suffix length to the performance of PAPA. *PAPA_M_N* denotes the segment structure with M prefix GOPs and N suffix GOPs. All the curves in the figure have the same prefix length of 5 GOPs. Small segment size leads to better performance at low cache percentage because of the significantly smaller early start time. A reasonable larger segment size is better when the cache percentage increases to the medium range due to the fast decline of waiting time and early start time. At high caching percentage, waiting time becomes zero for all suffix lengths as all prefixes are cached. Among all curves, *PAPA_Imp_5_5* has the smallest early start time on average and therefore performs the best.

D. Performance Comparison

In this section, we compare the performance of DECA with selected reference schemes. As the user satisfaction score is the most representative metric to evaluate the performance of the system, we will compare the different schemes according to their achievable final user satisfaction score.

As shown in the last section, the improved PAPA cache management scheme always performs the same or better than the original PAPA approach in [29]. Therefore, we use the improved PAPA scheme for comparison. Furthermore, as different segment structures (i.e., different suffix lengths) achieve the highest final score for different percentages of cached content, we pick the highest score at each percentage in the comparison, which forms an upper bound of PAPA.

We also compare DECA with cache management schemes which have no segment-prefix structure. In this case, each GOP



(a) L_P=$\frac{1}{2}L_S$ (b) L_P=5

Fig. 14. Performance of PAPA as a function of the percentage of cached content. (α_V=0.8, α=0.8)

is a possible starting point. Two approaches, on the frame level and on the GOP level will be compared. The End-GOP (EGOP) approach works on the frame level and drops frames evenly from the GOP with the lowest popularity towards the one with the highest popularity until the remaining part can be cached. Only when the last frames of all GOPs have been deleted, the second last frame in the GOP with the lowest popularity will be dropped. The Whole-GOP (WGOP) approach always deletes the whole GOP with the lowest popularity when not enough cache is available until all remaining GOPs can be held in the cache. These two strategies lead to the finest granularity but some initial delay unless the L_P successive GOPs from the requested point are available on the proxy. Early playout does not apply to these schemes. Similar as the segment-based schemes [14]–[16] introduced in Section I, the WGOP scheme also caches the video portions with the highest popularity, but with a much finer granularity. It supports both viewing modes, playing sequentially from the beginning or random access to video content. Obviously, WGOP should outperform the traditional segment-based approaches in [14]–[16]. Therefore, we use WGOP as one of the comparison schemes in this work.

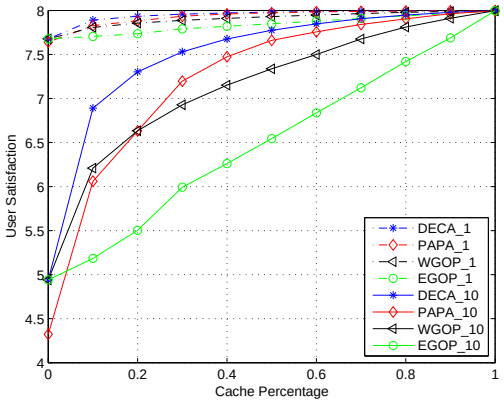
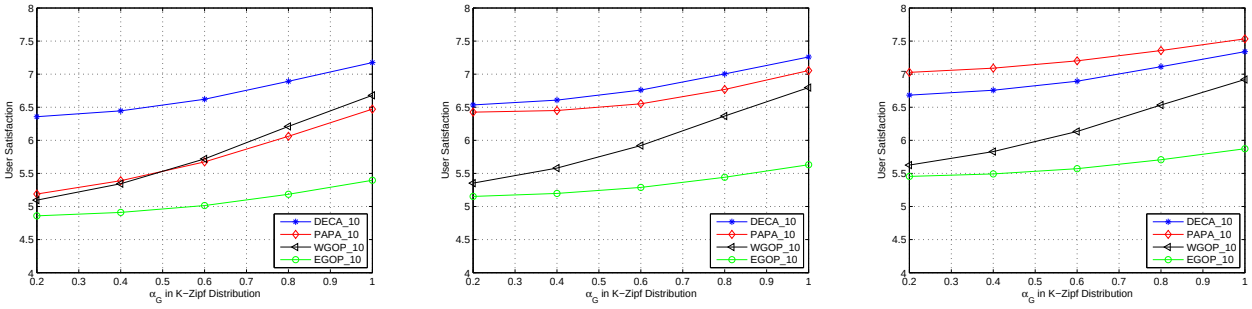


Fig. 15. User satisfaction for DECA and comparison schemes as a function of cache percentage. (α_V=0.8, α_G=0.8)

Fig. 15 shows the final user satisfaction score for the three comparison schemes. *PAPA_M* represents the upper bound of the improved PAPA. *WGOP_M* and *EGOP_M* denote the two no segment structure approaches, respectively. When the network condition between server and proxy is good (i.e., M=1), the four approaches achieve very similar performance as the initial delay for loading the unavailable requested data is very small, and most likely, "waiting" is the better mode. When the network condition is unfavorable (i.e., M=10), the performance gap between the different schemes becomes obvious. *DECA_10* performs the best at all cache percentages. The performance gain compared with *WGOP_10* is not so significant because for α_G=0.8 in the K-Transformed Zipf distribution, it gives strong emphasis to some GOPs and *WGOP_10* purely follows the popularity. Therefore, it pays high attention to popular GOPs by fully victimizing the GOPs with medium to low popularity. This is why at low cache percentage, it even performs better than PAPA. However, *PAPA_10* performs better than *WGOP_10* at middle to high cache percentage by enabling early start instead of pure waiting in WGOP. The *EGOP_10* approach considers the extreme fairness between GOPs, which wastes too much resources for the video content with low popularity. Without the help of early start, *EGOP_10* leads to the lowest overall performance. Please note, *PAPA_10* leads to the lowest user satisfaction score when nothing is cached (i.e., cache percentage is 0%). This is because that besides the same initial waiting time experienced by all approaches, PAPA has an additional early start according to its design which says that the playout should always start from the beginning of a segment.

Fig. 16(a), (b), (c) show the final score for DECA and the comparison schemes as a function of α at a cache percentage of 10%, 20% and 30%, respectively. DECA performs again the best for all cache percentages and all α_G values and achieves an even larger performance gain compared with Fig. 15 when α_G is small. The performance of WGOP decreases dramatically with the decrease of α_G because the difference of popularity between fragments is much smaller and a purely popularity oriented approach becomes less efficient. The per-



(a) Cache Percentage = 10% (b) Cache Percentage = 20% (c) Cache Percentage = 30%

Fig. 16. User satisfaction of DECA and comparison schemes as a function of α_G in the Zipf distribution.(α_V=0.8, L_P=10)

formance of PAPA improves significantly when the cache percentage increases, as the early start mode makes up the score loss caused by the even distribution of prefixes.

V. Conclusion

In this work we have presented our subjective tests for VoD services. We have observed that users prefer a small shift of the playout starting point rather than experiencing a noticeable initial delay. From the subjective test scores, we have derived mathematical models, which show the influence of waiting time and early start time on the user satisfaction. Based on that, a prefix-segment based partial caching algorithm DECA is proposed. It introduces a variable size segment structure and flexible serving modes and is able to approach the real popularity distribution and dynamically adjust the segment size according to the current situations. A two level proxy-caching framework is also introduced, where, the proposed DECA algorithm can be employed to improve the caching efficiency. Simulation results show significant performance improvements compared to related approaches.

References

[1] Z. Zhang, Y. Wang, D. H. C. Du, and D. Su, "Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 4, pp. 429–442, Aug. 2000.

[2] W.-H. Ma and D. H. C. Du, "Reducing bandwidth requirement for delivering video over wide area networks with proxy server," *IEEE Trans. Multimedia*, vol. 4, no. 4, pp. 539–550, Dec. 2002.

[3] S.-H. Chang, R.-I. Chang, J.-M. Ho, and Y.-J. Oyang, "An optimal cache algorithm for streaming VBR video over a heterogeneous network," *Journal of Computer Communications*, vol. 28, no. 16, pp. 1852–1861, Feb. 2005.

[4] S. Jin, A. Bestavros, and A. Iyengar, "Network-aware partial caching for internet streaming media," *Multimedia Systems*, vol. 9, no. 4, pp. 386–396, Oct. 2003.

[5] Z. Miao and A. Ortega, "Proxy caching for efficient video services over the internet," in *Proc. International Packet Video Workshop (PV'99)*, New York, NY, Apr. 1999.

[6] ——, "Scalable proxy caching of video under storage constrains," *IEEE J. Select. Areas Commun.*, vol. 20, no. 7, pp. 1315–1327, Sept. 2002.

[7] H. R. Oh and H. Song, "Scalable proxy caching algorithm minimizing client's buffer size and channel bandwidth," *Journal of Visual Communication and Image Representation*, vol. 17, no. 1, pp. 57–71, Feb. 2006.

[8] ——, "Metafile-based scalable caching and dynamic replacing algorithms for multiple videos over quality-of-service networks," *IEEE Trans. Multimedia*, vol. 9, no. 7, pp. 1535–1542, Nov. 2007.

[9] K. Li, K. Tajima, and H. Shen, "Cache replacement for transcoding proxy caching," in *Proc. IEEE International Conference on Web Intelligence (WI'05)*, Compiegne, France, Sept. 2005.

[10] W. Qu, K. Li, H. Shen, Y. Jin, and T. Nanya, "The cache replacement problem for multimedia object caching," in *Proc. IEEE International Conference on Semantics, Knowledge, and Grid (SKG'05)*, Beijing, China, Nov. 2005.

[11] C.-F. Kao and C.-N. Lee, "Aggregate profit-based caching replacement algorithms for streaming media transcoding proxy systems," *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 221–230, Feb. 2007.

[12] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE International Conference on Computer and Communications (INFOCOM'99)*, New York, NY, Apr. 1999.

[13] S. H. Park, E. J. Lim, and K. D. Chung, "Popularity-based partial caching for vod systems using a proxy server," in *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS'01)*, San Francisco, CA, Apr. 2001.

[14] K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," in *Proc. International Conference on World Wide Web (WWW'01)*, Hongkong, China, May 2001.

[15] ——, "Segmentation of multimedia streams for proxy caching," *IEEE Trans. Multimedia*, vol. 6, no. 5, pp. 770–780, Oct. 2004.

[16] S. Chen, H. Wang, X. Zhang, B. Shen, and S. Wee, "Segment-based proxy caching for internet streaming media delivery," *IEEE Multimedia*, vol. 12, no. 3, pp. 59–67, July 2005.

[17] R. Rejaie and J. Kangasharju, "Mocha: A quality adaptive multimedia proxy cache for internet streaming," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, Port Jefferson, NY, June 2001.

[18] J. Liu, X. Chu, and J. Xu, "Proxy cache management for fine-grained scalable video streaming," in *Proc. IEEE International Conference on Computer and Communications (INFOCOM'04)*, Hongkong, China, Mar. 2004.

[19] S. Chen, B. Shen, S. Wee, and X. Zhang, "SProxy: A caching infrastructure to support internet streaming," *IEEE Trans. Multimedia*, vol. 9, no. 5, pp. 1062–1072, Aug. 2007.

[20] M. Hofmann, T. E. Ng, K. Guo, S. Paul, and H. Zhang, "Caching techniques for streaming multimedia over the internet," Bell Labs, Holmdel, NJ, Tech. Rep. BL011345-990409-04TM, Apr. 1999.

[21] E. Balafoutis and I. Stavrakakis, "Proxy caching and video segmentation based on request frequencies and access costs," in *Proc. IEEE International Conference on Telecommunications (ICT'03)*, Tahiti, France, Feb. 2003.

[22] J. Z. Wang and P. S. Yu, "Fragmental proxy caching for streaming multimedia objects," *IEEE Trans. Multimedia*, vol. 9, no. 1, pp. 147–156, Jan. 2007.

[23] C. Griwodz, M. Bör, and L. C. Wolf, "Long-term movie popularity models in video-on-demand systems of the life of an on-demand movie," in *Proc. ACM International Conference on Multimedia*, Seattle, Washington, Nov. 1997.

[24] S. Acharya, B. Smith, and P. Parnes, "Characterizing user access to videos on the world wide web," in *Proc. ACM/SPIE Multimedia Computing and Networking (MMCN'98)*, San Jose, CA, Jan. 1998.

[25] G. K. Zipf, *Human Behaviour and the Principles of Least Effort*. Cambridge, MA: Addison-Wesley, 1949.

[26] H. Yan and D. K. Lowenthal, "Popularity-aware cache replacement in streaming environments," in *Proc. International Conference on Parallel and Distributed Computing Systems (PDCS'03)*, Reno, Nevada, Aug. 2003.

[27] J. Yu, C. T. Chou, X. Du, and T. Wang, "Internal popularity of streaming video and its implication on caching," in *Proc. IEEE International Conference on Advanced Information Networking and Application (AINA'06)*, Vienna, Austria, Apr. 2006.
[28] C. Zheng, G. Shen, and S. Li, "Distributed prefetching scheme for random seek support in peertopeer streaming applications," in *Proc. ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming*, Hilton, Singapore, Nov. 2005.
[29] L. Shen, W. Tu, and E. Steinbach, "A flexible starting point based partial caching algorithm for video on demand," in *Proc. IEEE International Conference on Multimedia and Expo (ICME'07)*, Beijing, China, July 2007.
[30] VideoLAN. VLC media player. [Online]. Available: http://www.videolan.org/vlc/
[31] Arizona State University. Video traces for network performance evaluation. [Online]. Available: http://trace.eas.asu.edu/

Wei Tu (S'04) received the B.S. from East China University of Science & Technology, Shanghai, China, in 1999 and the M.S. degree in electronic engineering from Technische Universität München, Munich, Germany, in 2003. He jointed the Media Technology Group, Institute of Communication Networks, Technische Universität München in 2004 and is currently working toward the Ph.D. degree. His research interests include error robust video transmission, RD-optimized scheduling and cache management for video on demand services.

Eckehard Steinbach (M'96-SM'08) studied electrical engineering at the University of Karlsruhe, Karlsruhe, Germany, the University of Essex, Colchester, U.K., and ESIEE, Paris, France. He received the Engineering Doctorate from the University of Erlangen-Nuremberg, Germany, in 1999.

From 1994 to 2000, he was a Member of the Research Staff of the Image Communication Group, University of Erlangen-Nuremberg. From February 2000 to December 2001, he was a Postdoctoral Fellow with the Information Systems Lab, Stanford University, Stanford, CA. In February 2002, he joined the Department of Electrical Engineering and Information Technology, Technische Universität München, Munich, Germany, as a Professor for Media Technology. His current research interests are in the area of audio-visual-haptic information processing, image and video compression, error-resilient video communication, and networked multimedia systems.

Muhammad Muhammad was born in Beirut, Lebanon on Feb. 25 1984. He received his MSc. in Communications Engineering from Technische Universität München, Munich, Germany, in 2007. Since April 2008 he has been with the Institute of Communications and Navigation at the German Aerospace Centre (DLR), Oberpfaffenhofen, Germany. His areas of current interest include higher layers FEC, ARQ techniques and Transport layer protocols design for satellite communications.

Xiaoling Li studied Information technology at Southeast University (China). From 2005 to 2007 he was a student of the MSCE (Master of Science in Communications Engineering) program at Technische Universität München (Germany), where he received his Master degree. In April 2008 he joined Fujitsu Microelectronics Europe as an Application Engineer dealing with Intellectual Property related aspects.