Efficient Dissemination of Personalized Video Content in Resource-Constrained Environments

Piyush Parate, Lakshmish Ramaswamy, Suchendra Bhandarkar, Siddhartha Chattopadhyay and Hari Devulapally

Abstract— **Video streaming on mobile devices such as PDA's, laptop PCs, pocket PCs and cell phones is becoming increasingly popular. These mobile devices are typically constrained by their battery capacity, bandwidth, screen resolution and video decoding and rendering capabilities. Consequently, video personalization strategies are used to provide these resource-constrained mobile devices with personalized video content that is most relevant to the client's request while simultaneously satisfying the client's resource constraints. Proxy-based caching of video content is a proven strategy to reduce both client latencies and server loads. In this paper, we propose novel video personalization server and caching mechanisms, the combination of which can efficiently disseminate personalized videos to multiple resource-constrained clients. The video personalization servers use an automatic video segmentation and video indexing scheme based on semantic video content. The caching proxies implement a novel cache replacement and multi-stage client request aggregation algorithm, specifically suited for caching personalized video files generated by the personalization servers. The cache design also implements a personalized video segment calculation algorithm based on client's content preferences and resource constraints. The paper reports series of experiments that demonstrate the efficacy of the proposed techniques in scalably disseminating personalized video content to resource constrained client-devices.**

Index Terms— **Video personalization; Caching; Cache replacement; Request aggregation**

I. INTRODUCTION

RECENT proliferation of mobile computing devices and networking technologies has created enormous opportunities for mobile device users to communicate with multimedia servers. As handheld mobile computing and communication devices such as personal digital assistants (PDAs), pocket-PCs and cellular devices have become increasingly capable of storing, rendering and display of multimedia data, the user demand for being able to view streaming video on such devices has grown considerably. For example, a mobile handheld client may be interested in viewing a video showing sports highlights or weather forecast video for his/her travel destination. One of the natural limitations of typical handheld mobile devices is that they are resource constrained, i.e., constrained by their battery capacity, screen resolution, video decoding and rendering capability, and, in many situations, by the available network bandwidth due to their mobility. Thus, the original video content often needs to be personalized in order to fulfill the client's request while simultaneously satisfying various client-side and system-level resource constraints. Numerous video personalization strategies have been developed [5], [6], [7] in order to provide these resource-constrained devices with personalized video content that is most relevant to the client's request and the available client-side and network resources.

Since mobile clients are typically present in remote network locations from the personalizing servers, it is desirable to intelligently cache portions of the video files at intermediate caching proxies. This would not only reduce latency observed at the client end, but would also enable local caches to share the loads on origin servers thereby enhancing the system scalability.

This paper presents the design and implementation of a novel framework for scalably disseminating personalized video content to large numbers of resource constrained client-devices. Our framework is characterized by the following two unique features.

1. Multiple video personalization servers (VPS) [15],[16] which perform automatic video segmentation and video indexing based on video content semantics
2. Multitude of caches which store video segments and perform on the fly composition of personalized video streams based on the clients' content preferences and resource constraints. Our video stream creation strategy is based upon a solution to the Multiple-Choice Multi-Dimensional Knapsack Problem [9],[10] (MMKP). The caches perform Multi-stage client request aggregation. They also incorporate a novel cache replacement policy that is sensitive to the existence of different personalized versions of same underlying video data.

The proposed system is evaluated through series of experiments. The results show that our techniques and mechanisms yield substantial performance benefits.

P. Parate, L. Ramaswamy, S. Bhandarkar and H. Devulapally are with Dept. of Computer Science, The University of Georgia, Athens, GA 30602, USA (e-mail: {parate, laks, suchi, hari}@ cs.uga.edu).

S. Chattopadhyay is with Google, Inc., Mountain View, CA 94043 (e-mail: siddhartha2k5@gmail.com).

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

II. OVERVIEW

In this section, we discuss the high-level architecture of our personalized video dissemination framework. As depicted in Figure 1, the environment consists of one or more *origin servers* that host video content and a number of *proxy caches*. Each proxy cache is responsible for clients in a certain geographical area in the sense that their requests to video content first reach the proxy cache. Upon receiving a client request, the proxy cache obtains respective video segments from the origin servers (if segments are not already available at the cache) composes the video stream and serves the client.

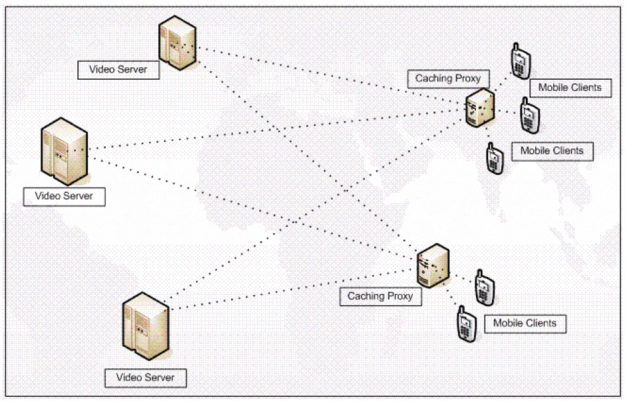


Figure 1. Multi-client Caching Proxies and Multiple Video Personalization Server architecture

Communication between clients and the caching proxy typically occurs in two phases, wherein the client first asks for the content to be generated and then downloads the content. Since the actual download occurs in the second phase of communication, the caching proxy serving as an intermediary between the clients and the VPS(s), can *learn* important statistics about the requests in order to pre-customize the video to be disseminated based on the download pattern that is to follow the first phase of communications.

Our framework supports multiple levels of membership such as "paying" and "non-paying". The cache design uses a content-aware video re-encoding algorithm termed as Features, Motion and Object Enhanced Multi-Resolution FMOE-MR video encoding [11], in order to create two versions of the original video file of lower visual quality and correspondingly lower file size. These versions are useful in delivering videos of different visual quality to different clients depending on their levels of membership. Before the client request is processed to generate the content response, the cache performs semantic aggregation of the client requests; further improving the cache replacement policy and hence reducing the average client latency.

As the caching proxy serve clients based on a two-phase communication, and has the ability to generate videos of different visual quality to serve multiple clients with different levels of membership, it performs significantly better than standard caching techniques.

In the next section, the design of VPS server is described, following which, we discuss the cache design and implementation of the proxy caches.

III. VIDEO PERSONALIZATION SERVER

The videos hosted by the video personalization server (VPS) are first segmented and indexed. The videos are then transcoded at multiple levels of abstraction based on their content. This allows for video personalization based on clients' preferences and resource constraints.

A. Video Segmentation and Indexing

A stochastic multi-level Hidden Markov Model (HMM)-based algorithm is used for video segmentation and indexing wherein the input video stream is classified frame by frame into semantic units [17]. A semantic unit within a video stream is a video segment that can be associated with a clear semantic meaning or concept, and consists of a concatenation of semantically and temporally related video shots or video scenes. Instead of detecting video shots or scenes, it is often much more useful to recognize semantic units within a video stream to be able to support video retrieval based on high-level semantic content. Note that visually similar video shots or video scenes may be contained within unrelated semantic units. Thus, video retrieval based purely on detection of video shots or video scenes will not necessarily reflect the semantic content of the video stream.

The semantic units within a video stream can be spliced together to form a logical video sequence that the viewer can understand. In well organized videos, such as TV broadcast news and sports programs, the video can be viewed as a sequence of semantic units that are concatenated based on predefined video program syntax. Parsing a video file into semantic units enables video retrieval based on high-level semantic content and playback of logically coherent blocks within a video stream. Automatic indexing of semantic components within a video stream can enable a viewer to jump straight to points of interest within the indexed video stream, or even skip advertisement breaks during video playback.

In the proposed scheme, a video stream is modeled at both, the semantic unit level and the program model level. For each video semantic unit, an HMM is generated to model the stochastic behavior of the sequence of feature emissions from the image frames. Each image frame in a video stream is characterized by a multi-dimensional feature vector. A video stream is considered to generate a sequence of these feature vectors based on an underlying stochastic process that is modeled by a multi-level HMM.

In this work, we extract two categories of features from each image frame in the video stream. The first category includes a set of simple features. The dynamic characteristics of the image frames comprising the video stream are captured by the differences of successive image frames at both, the pixel level and the histogram level. Various motion-based measures describing the movement of the objects in the image frames are used, including the motion centroid of the image,

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

and intensity of motion. Measures of illumination change at both, the pixel level and the histogram level are also included in the multi-dimensional feature vector. Definitions of these features are given in [2]. In the second feature category, Tamura features [20] are used to capture the textural characteristics of the image frames at the level of human perception. Tamura contrast, Tamura coarseness and Tamura directionality have been used successfully in content-based image retrieval [20]. In our work, inclusion of these features is observed to improve the accuracy of temporal video segmentation and video indexing.

In the proposed video segmentation and video indexing scheme based on semantic video content, we define six semantic concepts for TV broadcast news video, i.e. *News Anchor*, *News*, *Sports News*, *Commercial*, *Weather Forecast* and *Program Header*, and three semantic concepts for Major League Soccer (MLS) video, i.e. *Zoom Out*, *Close Up* and *Replay*. An HMM is formulated for each individual semantic concept. The optimal HMM parameters for each semantic unit are learned from the feature vector sequences obtained from the training video data. In the proposed scheme, the HMMs for individual semantic units are trained separately using the training feature vector sequences. This allows for modularity in the learning procedure and flexibility in terms of being able to accommodate various types of video data. In our work, we adopt a universal left-to-right HMM topology, i.e., an HMM topology where no backward state transitions are allowed, with continuous observations of the feature vector emissions. The distribution of the feature vector emissions in the HMM is approximated by a mixture of Gaussian distributions.

The search space for the proposed single-pass video segmentation and video indexing procedure is characterized by the concatenation of the HMMs corresponding to the individual semantic units. The HMM corresponding to an individual semantic unit essentially models the stochastic behavior of the sequence of image features within the scope of that semantic unit. Transitions amongst these semantic unit HMMs are regulated by a pre-specified video program model. The Viterbi algorithm is used to determine the optimal path in the concatenation of the HMMs in order to segment and index video stream in a single pass [17].

IV. Video Personalization Cache

In the previous section, we have detailed the design of a video personalization server (VPS) that receives requests from multiple clients, and creates personalized videos for them. The cache serves as an intermediary between the clients and the VPS (Figure 2). In the following subsections, we first detail the design of the proposed cache. Then, we describe the working of multi-stage client request aggregation in detail. Next, we describe how the proposed cache creates different versions of the video files using content-aware video processing. Finally, we describe in detail the protocol followed by the cache and clients in order to synchronize with the VPS.

A. Cache Design

The proposed multiple caching proxies serve the following purposes:

a) They act as buffer between multiple VPS(s) and multiple mobile clients, in order to reduce the overall latency observed by the clients.
b) They simulate generative proxy video servers that generate videos of different visual quality in order to increase efficiency of the cache replacement policy.
c) They perform semantic client request aggregation to reduce the request processing load and hence increase the cache replacement performance.

B. Personalized Video Segment Calculation

1) Relevance Value of a Video Segment and Summary

Video segments are indexed using semantic terms. Each video segment is assigned a relevance value based on the client's preference with regard to video content. Assume video segment S_i is indexed by a semantic term T_i. In its request, the client specifies a preference for video content using a descriptive term labeled as P. The relevance value V_i assigned to the video segment S_i is then given by:

$$V_i = similarity(T_i, P), 0 \le V_i \le 1 \quad (1)$$

In the current implementation the similarity is evaluated using the lch semantic similarity measurement algorithm [3].

Each indexed video segment is summarized at multiple levels of abstraction using content-aware key frame selection and motion panorama computation algorithms [15]. Each video summary consists of a set of key frames and motion panoramas. For each video segment, its original version is assumed to contain the greatest amount of detail; whereas its summary at the highest level of abstraction is assumed to contain the least amount of detail. It is reasonable to assume that the amount of information contained within a video summary (relative to original version) is related to its duration, i.e.,

$$v_i = v_{i0} \cdot f(L_i / L_0) \quad (2)$$

where, v_{i0} is the relevance value of the original video segment, L_0 and L_i are the time durations of the original video segment and the video summary respectively. Typically, the amount of information contained within a video summary (relative to original version) does not necessarily increase linearly with its relative duration.

2) MMKP-based Video Personalization

The objective of video personalization is to present a customized or personalized video summary that retains as much of the semantic content desired by the client as possible, but within the resource constraints imposed by the client. The client typically wants to retrieve and view only the contents that match his/her content preferences. In order to generate the

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

personalized video summary, the client preferences, the client usage environment and client-side and system-level resource constraints need to be considered. The personalization engine selects the optimal set of video contents (i.e., the most relevant set of video summaries) for the client within the resource constraints imposed by the client. In our work we are implementing the MMKP-based video personalization strategy to generate a customized response to the client's request while satisfying multiple client-side and system-level resource constraints. Compared to the 0/1 Knapsack Problem (KP)-based and the Fractional Knapsack Problem (FKP)-based video personalization strategies presented in [5], [6] and [7], MMKP-based video personalization strategy is shown to include more relevant information in its response to the client's request. The MMKP-based personalization strategy is also shown to support multiple client-side constraints, in contrast to the 0/1KP-based and the FKP-based personalization strategies which can support only a single client-side resource constraint at a time.

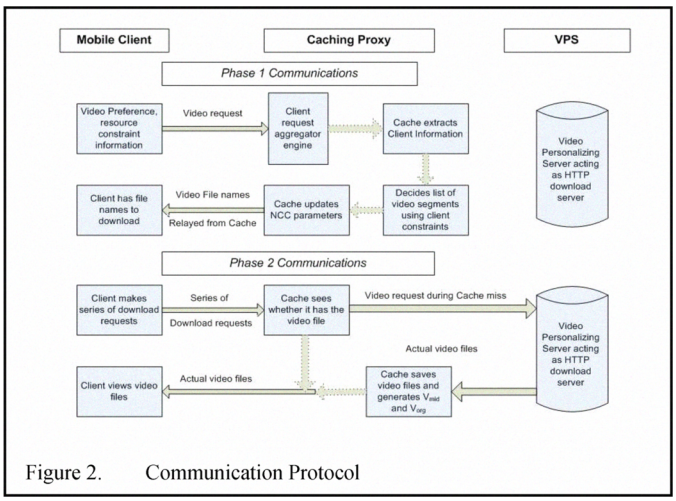


Figure 2. Communication Protocol

In many applications, it is desirable to provide the client with as much information as possible. In such cases it may be preferable to include two shorter video summaries in the response rather than a single video segment of longer duration that contains more details. For example, if a client needs to browse the sports news of the day, it might be helpful to provide him/her with multiple, though short, sports news summaries rather than a single long and detailed video segment containing news of a specific sport. The implemented Multiple-Choice Multi-Dimensional Knapsack Problem (MMKP)-based video personalization strategy [8], [9] performs around the above mentioned technique.

C. Multistage Client Request Aggregation

The client requests arriving at the caching proxy can be distinguished based on; arrival time, video content preference and client-side resource constraints. This implementation uses a novel approach to reduce the content generation time at the caching proxy by semantically grouping the client requests using the above mentioned features. The grouping of several client requests into one request increases the proxy cache hit percentage and eventually reduces the average latency at the client-side. The semantic aggregation is performed in three stages. The first stage involves batching client requests arriving within a pre-specified time window. The resulting group is termed a batch group. In the second stage, client requests within a batch group with similar video content preferences are clustered into batch service groups. Let q denote the ordered set of semantic terms used to index video segments, and Q denote the ordered set of semantic terms that the clients can use in their requests to specify their video content preferences, such that $q \subseteq Q$. Also assume that a similarity matrix $S_{size(q)\ x\ size(Q)}$ used to define the similarity of semantic terms, where $0 \le S_{ij} \le 1$ represents the semantic similarity of term $t_i \in q$ and term $t_j \in Q$. The value of S_{ij} is computed using the *lch* algorithm described in [3]. Let T be the semantic term used in the client's video content preference list, and let k be the index of term T in the ordered set Q, then the client *content preference vector*

$$P_c = (s_{1k}, s_{2k}, \ldots s_{size(q)k}) \quad (3)$$

where, S_{ik} , $1 \le i \le size(q)$ is the semantic similarity between term $t_i \in q$ and $t_k \in Q$, and can be obtained from the similarity matrix $S_{size(q)\ x\ size(Q)}$. The *preference clustering* algorithm uses the cosine similarity measure between a pair of client query content preference vectors P_{c1} and P_{c2} to represent the distance between them. The *k-means* clustering algorithm is used to cluster client requests with similar content preference values into a group, termed as a *content service group*. The client requests within a *content service group* are further clustered based on client-side system-level resource constraints to generate a set of *service groups*. In our experiments, client requests with similar values for the video viewing time limit are clustered together using the *k-means* clustering algorithm.

D. Creating Different Layers of the Video

From each video segment that the cache receives for storage, two additional versions, or layers, of the same video are created. The original video segment, or layer, is designated as V_{org}. V_{mid} is an intermediate layer video of lower visual quality (and smaller file size) than V_{org} whereas V_{base} is the base layer video of lowest visual quality and smallest file size. The lower quality video layers V_{mid} and V_{base} are used to design an efficient cache replacement policy.

The different video layers are essentially transcoded versions of the original video at different levels of visual quality with file sizes that are significantly smaller than the file size of the original video. We have experimented with novel transcoding methods such as Features, Motion and Object Enhanced Multi Resolution (FMOE-MR) video encoding [11] and Ligne-Claire video encoding [18]. Due to its relative simplicity in terms of implementation for the purpose of caching, we have chosen FMOE-MR as the

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

transcoding technique for this paper. In the following subsections, we describe briefly how the two layers, V_{mid} and V_{base}, are created from the original video segment V_{org}.

1) Creating V_{mid}

The video layer V_{mid} represents an intermediate-level video which has a more compact representation than the original video V_{org}, albeit at the cost of lower visual quality. The video layer V_{mid} is generated using a novel multi-resolution video encoding technique termed as Features, Motion and Object-Enhanced Multi-Resolution (FMOE-MR) video encoding [11].

The FMOE-MR video encoding scheme is based on the fundamental observation that applying a low pass filter in the image color space is equivalent to DCT coefficient truncation in the corresponding DCT (frequency) space [12].

2) Generating V_{base}

The base video layer V_{base} is generated by a method similar to the Gaussian smoothing performed in the case of FMOE-MR video encoding. The primary difference is that, in the case of the V_{base} video layer, the smoothing operation is performed uniformly over the entire video frame in contrast to FMOE-MR video encoding where the extent of smoothing can vary within a video frame based on the perceptual significance of the region under consideration. This results in further dramatic decrease in the file size, albeit at the loss of video quality. Note that V_{base} is of much lower visual quality than V_{mid} since object-based enhancement is not used.

3) File Size Overload

V_{mid} and V_{base} are additional files stored in the cache. The total sum of all the files V_{org}, V_{mid} and V_{base} is around 1.5 times that of V_{org}; i.e., the combined size of V_{mid} and V_{base} adds $\approx 50\%$ overhead to storage space required for storing the video segment.

Thus ensuring reasonably good visual quality for V_{mid} and V_{base} makes it possible to discard V_{org} to allow for extra space in the cache, and yet have videos of reasonable visual quality reside in the cache, during a cache replacement.

E. Cache Replacement

A novelty in the proposed cache design is that fact that a hit or a miss score depends on the *client-type*. A *client-type* defines the membership status of the client with regard to the video personalization service. Without loss of generality, we have used two categories of clients - either they are *paying* or *subscribing* clients, who are paying for the best quality of service, or they are *non-paying* clients who receive videos of varying quality depending on the availability of the videos in the cache. For non-paying clients, the best quality video is not guaranteed; however the best quality video is guaranteed for the paying client.

If the requesting client is *non-paying*, then the cache checks if the requested file V_{org} is present in the cache. If it does not exist in the cache, it checks whether a lower quality version of the file, i.e., V_{mid}, is present in the cache. If V_{mid} is absent, then it checks whether V_{base} is present in the cache. If neither V_{mid} nor V_{base} are present in the cache the client request is treated as a cache miss. In contrast, if the requesting client is *paying* then the client request is treated as a cache miss if the requested file V_{org} is absent in the cache. Thus, for a *paying* client, the best quality video is provided, whereas for the non-paying client, the quality of video which is present in the cache is present; it might not be the best quality one.

In the event of a cache miss, whether the client type is a *paying* or *non-paying*, the requested file is relayed from the VPS, and also stored in the cache simultaneously. If there is not enough space available in the cache, a cache replacement policy is enforced to replace existing file(s) in the cache in order to make space for the requested file. If the cache cannot accommodate the requested file, an existing file with the minimum *retention-value* (RV) is discarded. The *retention-value* is essentially a number associated with each file in the cache; the lower the *retention-value*, the less valuable is the file to the cache. Thus, if an existing file is to be discarded from the cache in order to make space in the cache, the one with the lowest *retention-value* is removed.

The computation of the *retention-value* is specific to a cache replacement policy. For the proposed cache, we compute the *retention-value* as follows: *retention-value* = NCC/*fileSize,* where NCC is the *number of common clients* requesting that particular file and *fileSize* is the size of the file. This cache replacement policy has been termed NCCS (Number of Common Clients - Size) algorithm. NCCS essentially denotes the number of client requests for this particular file normalized by the file size. In order to assign higher priority to files which are requested by paying clients, the number of clients (NCC) is incremented by 10, instead of 1, for paying clients as opposed to non-paying clients. This information is obtained during the first phase of the communication between the cache and the VPS.

As will be seen in the following experimental results section, the proposed NCCS cache replacement policy that uses the proposed *retention-value* as the replacement criterion, is better suited for caching of personalized video than other standard cache replacement algorithms.

Recall that in the case of a miss, the file with the least *retention-value* in that category is discarded. The file replacement process is as follows. First, the video file with the minimum *retention-value* is identified. After that, an attempt is made to remove the original video segment; i.e., V_{org} corresponding to that video file. If V_{org} had already been removed previously, then V_{mid} is removed; if V_{mid} has also been removed, then V_{base} is removed. This top down approach ensures that the layer which takes up the largest amount of space in the cache is removed, so that space for more popular video segments, can be made.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

V. CLIENT-CACHE-SERVER COMMUNICATION

The client-cache-server communication is done in two phases:

Phase 1:

Multiple clients send their queries or requests to the caching proxy, which uses multi-stage client request aggregation algorithm to generate a sub-request of incoming client requests. Each request has, video preference essentially consisting of one category out of six possible categories (*News Anchor*, *News*, *Sports News*, *Commercial*, *Weather Forecast* and *Program Header*), the available battery time in the client device and bandwidth capacity as resource constraints. The cache generates a list of video segments based on the client video preference and resource constraints and sends the names of the video segments to be downloaded (in a specified order), as metadata back to each client.

Phase 2:

Each client now makes a series of requests to the caching proxy for files belonging to one of the four categories. The files can be streamed, or can be progressively downloaded, depending on the type of service available at the caching proxy. In either case, recall that the names of the files, and the order they appear in, have been specified in the metadata obtained in *Phase 1* of the communication.

VI. EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed caching proxy. First, we describe the experimental setup, and then discuss the results.

A. Experimental Setup

Our simulation environment consists of 1 video personalization server and three proxy caches. The server and caches were deployed on Pentium-4, 2.79 GHz Machines with 2 GB of RAM running on Linux Red Hat 4 Operating Systems. The number of clients varied between 50 and 200. However, unless specified otherwise, the results presented in this paper are for simulations involving 150 clients. It is assumed that around 40% of the clients are paying clients; the rest are non-paying. The time-window size for the first stage of client-request aggregation at the caching proxies was set to 3000ms.

Our experiments involved multiple sessions of client-cache-server communication. Each session consists of multiple clients, each performing the preference-request once. During each session the clients contact the HTTP server proxy with time lag determined by a random function generator which follows Poisson distribution with a max-delay of 4 seconds and variance of 2 seconds. The client preference value is generated using a random uniform distribution generator over the defined client preference vector. The resource constraints namely viewing-time limit and bandwidth are generated using Normal Distribution generators. In order to simulate mobile devices with different viewing time limits, the values for viewing time limits are modeled as a mixture of 2 normal distributions $N_1(\mu_1, \sigma_1^2)$ and $N_2(\mu_2, \sigma_2^2)$ where, $\mu_1 = 50$ seconds, $\sigma_1 = 70$ and $\mu_2 = 150$ seconds, $\sigma_2 = 70$ seconds. The Normal Distribution generator for bandwidth, $N_3(\mu_3, \sigma_3^2)$ has $\mu_3 = 180$ KBps and $\sigma_3 = 40$.

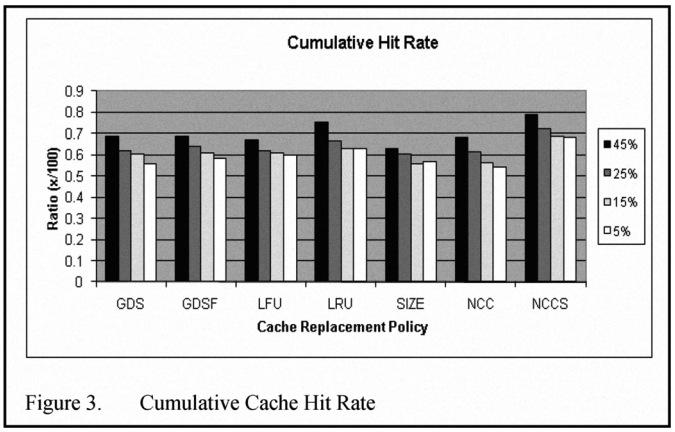


Figure 3. Cumulative Cache Hit Rate

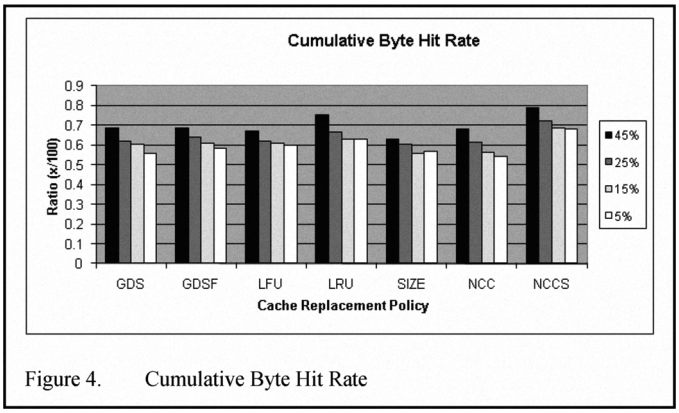


Figure 4. Cumulative Byte Hit Rate

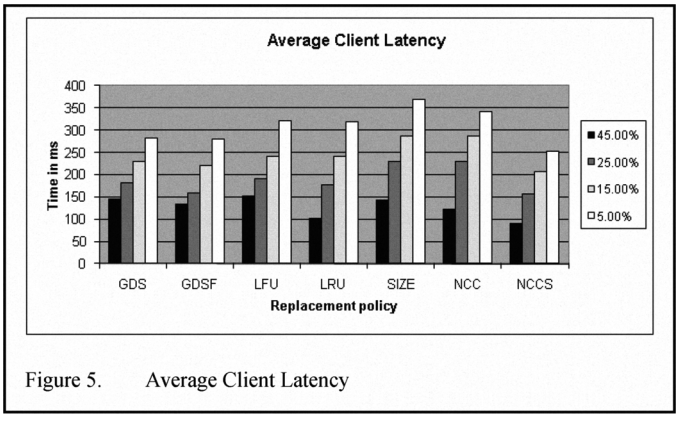


Figure 5. Average Client Latency

We use the standard metrics for evaluating cache performance, namely, *Hit Ratio*, *Byte Hit Ratio* and *Client-side Latency*. For completeness, we provide their definitions below:

Hit Ratio: The number of requests satisfied by the proxy cache as a percentage of total requests.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

Byte Hit Ratio: The number of bytes that transferred from the proxy cache as a percentage of total number of bytes transferred for all the requests.

Client-side Latency: The client latency is calculated as the time lapsed after the client makes the request during first phase of communication and when it receives the first byte of first video requested during the second phase of communication.

We have considered four other standard cache replacement policies in order to do a performance comparison with the proposed NCCS cache replacement algorithm. The replacement algorithms, with their corresponding *retention-functions*, are as follows:

GDS (Greedy-Dual-Size): The GDSF cache replacement algorithm [13] uses a retention value given by.

retention_value = Latency/ Size

In addition to removing files with the minimum retention value during a cache replacement, the GDS algorithm also subtracts this minimum *retention-value* from the *retention-value* for each of the other files in the cache. When a file is scored as a hit, the original *retention-value* of the file is recomputed.

GDSF (Greedy-Dual-Size-Frequency): The GDSF cache replacement algorithm [14] uses a retention value given by.

retention_value = nHits × Latency/ Size

Similar to GDS, GDSF subtracts this minimum *retention-value* from all the *retention-value* in each of the other files in the cache. When a file is hit, the original *retention-value* of the file is recomputed.

LRU (least Recently Used): The retention value is given by: *retention-value* = hitTime

This simple retention value is used to replace files that were accessed farthest in the past.

LFU (Least Frequently Used): The retention value is given by: *retention-value = nHits + nMisses*

In other words, the *retention-value* is the number of accesses to the file, which can be simply used as a measure of the frequency of access.

B. Simulation Results

In the first experiment, the origin server contains 48 video files. It is assumed that each file is 3000 KB. The number of clients in the environment is 150. We vary the sizes of the proxy caches from 5% to 45% of the storage available on the origin server. The results are shown in Figures 3, 4, and 5. Figure 3 displays the hit rates of various cache replacement strategies at various cache sizes. Figure 4 shows the corresponding byte hit ratios. As figures show, NCCS yields the highest hit rates at all cache sizes. The improvements in cache hit rates translate into reductions in client latencies as depicted in Figure 5.

In the second experiment, the VPS contains 41 files. However, unlike the first experiment, the sizes of files vary and are drawn uniformly from the range (2200 KB, 5500 KB). The rest of the configuration remains identical to the first experiment. Figure 6, Figure 7 and Figure 8 show the results of the experiments. Again we see NCCS performs best in terms of all three metrics.

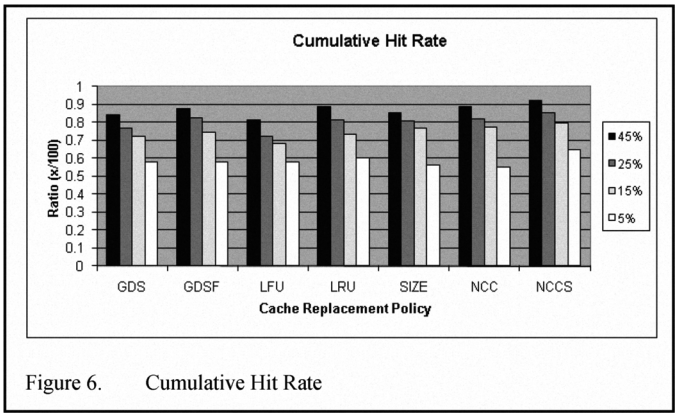


Figure 6. Cumulative Hit Rate

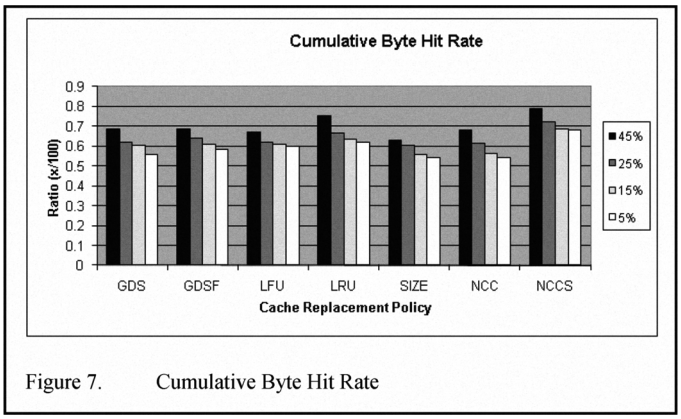


Figure 7. Cumulative Byte Hit Rate

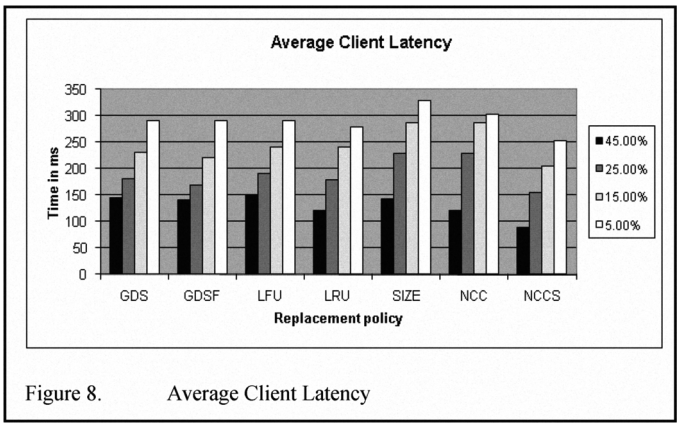


Figure 8. Average Client Latency

We surmise that the reason proposed NCCS cache replacement policy outperforms other cache replacement schemes is because of the fact that, in this particular case, due to the two-phase communication initiated between the clients and VPS, the NCCS scheme knows in advance which files will be asked for by the clients. Note that this is a special situation that can arise only in the case of the above two-phase communication. Thus, the cache captures the popularity of a file by computing the commonality of that particular file (video segment) amongst all the clients who will request the file in the near future. This fact is captured by the NCC statistics (number of common clients). In addition, the Size factor in the NCCS cache replacement policy makes NCCS prefer more, smaller files, compared to fewer, larger files,

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

which should be the case for a good caching scheme. As a result, the proposed NCCS cache replacement policy is observed to be the most successful.

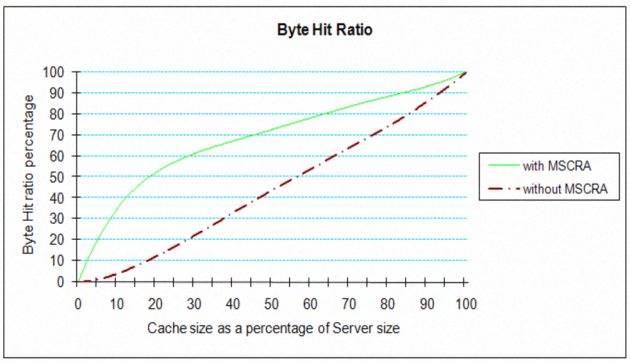


Figure 9. The Byte Hit Ratio calculated as average of all the caching proxies.

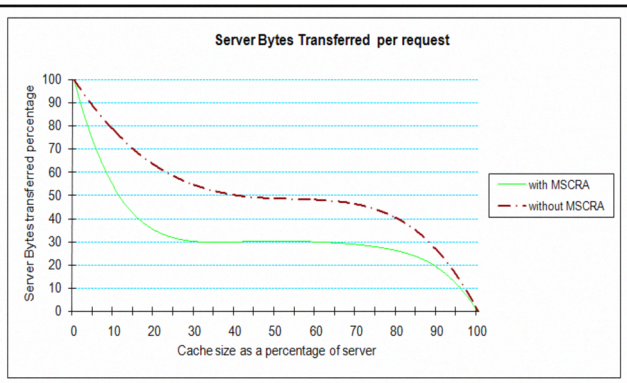
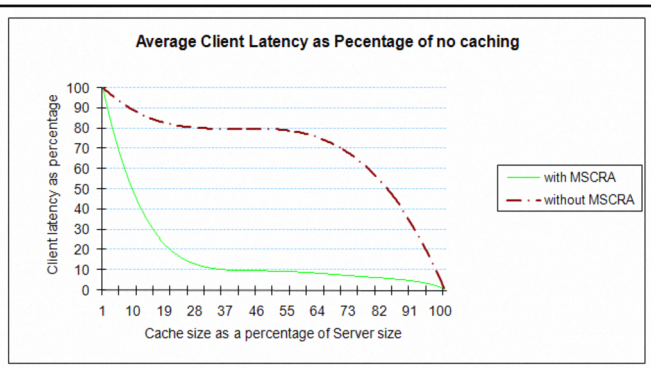


Figure 10. *The Server Bytes transferred per request calculated as a percentage of no-caching implementation.*



11. *The average client latency calculated as the percentage of no-caching implementation with increasing total size of caching proxies as a percentage of total size of VPSs.*

In the next set of experiments, we study the benefits of the multi-stage client request aggregation (MSCRA) technique. The simulation setup comprises of 150 clients, 41 files at the origin server with file sizes varying in the range (2200 KB, 5500 KB). We compare the scenarios wherein the proxy caches implement NCCS replacement strategy with and without MSCRA technique. Figure 9 shows the byte hit rates of the two cases when cache size varies from 0% to 100% of the storage at the origin server. Figure 10 shows the bytes transferred from the origin server and Figure 11 depicts the average client latency values. The results demonstrate that MSCRA results in substantial reduction in server loads and client latencies.

As a side note, we also observed that a cache running on a 2.0 GHz Pentium Processor with 2 GB RAM and 2 MB L2 cache, could create V_{mid} and V_{base} in real time; i.e., around 30 frames per second.

VII. Conclusion

Server-Cache architecture has been proposed and implemented, which can disseminate personalized video contents to resource-constrained devices. The proposed video personalization server (VPS) performs automatic video segmentation and video indexing based on semantic video content. A novel caching mechanism, dedicated to cache video segments generated by VPS, implementing the Multi-Dimensional Knapsack Problem [8],[9] (MMKP)-based video personalization strategy to calculate personalized video segments based on client preference and resource constraints has been proposed. The proposed caching mechanism uses an ingenious multi-stage client request algorithm and exploits the commonality of files across clients to use a novel cache replacement mechanism termed as NCCS (Number of Common Clients Size). The multi-stage client request algorithm uses semantic knowledge to group client requests and helps reduce the request load on the VPS as well as increase the efficiency of replacement algorithm. The cache also generates two lower quality versions of each video file. These versions aid in the dissemination of video files to multiple clients entitled to different levels of service based on whether they are paying for the video service or not. Results show that the proposed cache implementation using multi-stage client request aggregation, in conjunction with the proposed NCCS cache replacement policy, outperforms standard cache implementations. Thus, the video-cache architecture is suitable for personalized video dissemination to resource-constrained mobile devices such as mobile phones, PDAs, Pocket PCS and laptop computers operating in battery mode.

References

[1] Sikora, T., "Trends and perspectives in image and video coding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 6-17, Jan. 2005.

[2] Eickeler, S., and Müller, S. 1999. Content-based Video Indexing of TV Broadcast News using Hidden Markov models, *Proc. Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Processing*, March 1999, 2997-3000.

[3] Leacock, C., and Chodorow, M., Combining Local Context and WordNet Similarity for Word Sense Identification. in *WordNet: An Electronic Lexical Database*, C. Fellbaum (Editor), MIT Press, Cambridge, MA, 1998, 265-283.

[4] Wheeler, E.S., Zipf's Law and Why it Works Everywhere. *Glottometrics*, vol.4(2002), 45-48.

[5] Merialdo, B., Lee, K.T., Luparello, D., and Roudaire, J., Automatic Construction of Personalized TV News Programs. *Proc. ACM Conf. Multimedia*, (Orlando, FL, Sept. 1999), 323—331.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349

[6] T. S. Eugene Ng and Hui Zhang, "Towards Global Network Positioning", *Extended Abstract, ACM SIGCOMM Internet Measurement Workshop 2001*, San Francisco, CA, November 2001

[7] T. S. Eugene Ng and Hui Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches", *INFOCOM'02*, New York, NY, June 2002

[8] Tseng, B.L., and Smith, J.R., "Hierarchical Video Summarization Based on Context Clustering," *Proc. SPIE*, Vol. 5242(Nov. 2003), 14-25.

[9] Tseng, B.L., Lin, C.Y., and Smith, J.R., Video Personalization and Summarization System for Usage Environment. *Jour. Visual Communication and Image Representation*, Vol. 15(2004), 370–392.

[10] Akbar, M.D., Manning, E.G., Shoja, G.C., and Khan, S., Heuristic Solutions for the Multiple-Choice Multi-dimension Knapsack Problem. *Proc. Intl. Conf. Computational Science*, 2001, 659-668.

[11] Hernandez, R.P., and Nikitas, N.J., A New Heuristic for Solving the Multichoice Multidimensional Knapsack Problem. *IEEE Trans. System, Man and Cybernetics*, Part A, Vol. 35, No. 5(2005), 708-717.

[12] Vanderbei, R. J., *Linear Programming: Foundations and Extensions*, Kluwer Academic Publishers, 1997.

[13] Chattopadhyay, S., Luo, X., Bhandarkar, S. M. and Li, K., "FMOE-MR: content-driven multi-resolution MPEG-4 fine-grained scalable layered video encoding", *Proc. ACM Multimedia Computing and Networking Conference (ACM MMCN' 07),* San Jose, California, January, pp. 650404-1- 11, 2007.

[14] Geusebroek, J.-M., Smeulders, A. W. M. and Van De Weijer, J., "Fast Anisotropic Gaussian Filtering", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, March 2001.

[15] Jin S. and Bestavros, A., "Popularity-aware greedy dual-size web proxy caching algorithms", *Proc. 20th IEEE Intl. Conf. Distributed Computing Systems (ICDCS),* Taipei, Taiwan, Apr. 2000, pp. 254-261.

[16] Cherkasova, L., "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy", HP Laboratories Report No. HPL-98-69R1, April, 1998.

[17] Wei, Y., Bhandarkar, S.M. and Li, K., "Client-centered Multimedia Content Adaptation", *ACM Trans. Multimedia Computing, Communications and Applications (ACM TOMCCAP)* , in press.

[18] Wei, Y., Bhandarkar, S.M. and Li, K., "Video Personalization in Resource-Constrained Multimedia Environments", *Proc. ACM Conf. Multimedia*, Augsburg, Germany, Sept. 2007, pp. 902-911.

[19] Wei, Y., Bhandarkar, S.M. and Li, K., "Semantics-based Video Indexing Using a Stochastic Modeling Approach", *Proc. IEEE Intl. Conf. Image Processing (ICIP)*, San Antonio, TX, Sept. 2007.

[20] Chattopadhyay, S., Bhandarkar, S.M. and Li, K., "Ligne-Claire Video Encoding for Power Constrained Mobile Environments", *Proc. ACM Conf. Multimedia*, Augsburg, Germany, Sept. 2007, pp. 1036-1045.

[21] Chattopadhyay, S., Bhandarkar, S.M. and Li, K., "A Framework for Encoding and Caching of Video for Quality Adaptive Progressive Download", *Proc. ACM Conf. Multimedia*, Augsburg, Germany, Sept. 2007, pp. 775-778.

[22] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. 1995. Query by Image and Video Content: The QBIC System, *IEEE Computer Magazine*, September, 23 – 32.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8349
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8349