Collaborative Caching for Efficient Dissemination of Personalized Video Streams in Resource Constrained Environments

Suchendra Bhandarkar
Dept. of Computer Science
The University of Georgia
Athens, GA 30602-7404, USA
suchi@cs.uga.edu

Lakshmish Ramaswamy
Dept. of Computer Science
The University of Georgia
Athens, GA 30602-7404, USA
laks@cs.uga.edu

Hari K. Devulapally
Dept. of Computer Science
The University of Georgia
Athens, GA 30602-7404, USA
dhari28@gmail.com

ABSTRACT

The ever increasing deployment of broadband networks and simultaneous proliferation of low cost video capturing and multimedia enabled mobile devices have triggered a wave of novel mobile multimedia applications, resulting in the development of large scale systems for delivery of video streams to heterogeneous resource constrained mobile clients. Invariably, the video streams need to be personalized to provide a resource constrained mobile device with video content that is most relevant to the client's request while simultaneously satisfying the client-side and system-wide resource constraints. In this paper we present the design and implementation of a distributed system, consisting of several geographically distributed video personalization servers and proxy caches, for efficient dissemination of personalized video in a resource constrained mobile environment. With the objective of optimizing cache performance, a novel cache replacement policy and multi-stage client request aggregation strategy, both of which are specifically tailored for personalized video content, are proposed. A novel latency-biased collaborative caching protocol based on counting Bloom filters is designed for further enhancing the scalability and efficiency of disseminating personalized video content. The benefits and costs associated with collaborative caching for disseminating personalized video content to resource constrained and geographically distributed clients are analyzed and experimentally verified. The impact of different levels of collaboration amongst the caches and, the advantages of using multiple video personalization servers with varying degrees of mirrored content on the efficiency of personalized video delivery, are also studied. Experimental results demonstrate that the proposed collaborative caching scheme, coupled with the proposed personalization-aware cache replacement and client request aggregation strategies, provides a means for efficient dissemination of personalized video streams in resource constrained environments.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Multimedia Information Systems; H.5.1 [**Multimedia Information Systems**]: Video—*performance measures*

General Terms

Design, Measurement, Performance

Keywords

Collaborative Caching, Video Personalization, Cache Replacement, Request Aggregation

1. INTRODUCTION

Recent years have witnessed an ever increasing deployment of broadband networks accompanied by a simultaneous proliferation of multimedia-enabled mobile computing and communication devices such as netbook computers, tablet computers and smart cellular phones that have become increasingly capable of storing, rendering and displaying multimedia content. Consequently, user demand for web-based and cellular services that entail transmission and rendering of streaming video to/on these devices has grown considerably. Despite recent technological advances, networked environments consisting of mobile devices pose significant challenges on account of the inherent inhomogeneity arising from the varying client-side and system-wide resource constraints, the various client-initiated queries for information, the geospatial distribution and varying dynamic trajectories of the mobile clients, and varying client-side and server-side security and privacy requirements. The client-side resource constraints typically include limits on the battery capacity, screen resolution, and the video decoding and rendering capabilities of the mobile device whereas the most common system-wide resource constraint is the limited network bandwidth. Consequently, for effective transmission and dissemination of video in networked mobile environments it is imperative that the original video content be adapted or personalized in order to fulfill multiple client requests while simultaneously satisfying the various client-side and system-wide resource constraints, client-side and server-side security and privacy requirements and the constraints imposed by the geospatial distribution and dynamic trajectories of the mobile clients relative to the server(s).

Conventional video adaptation or personalization techniques focus on adapting the bit rate, frame resolution and frame rate of the video stream to match the client-side and system-wide resource constraints. These adaptation techniques are based primarily on statistical analysis of low-level (i.e., pixel-level or feature-level) video content and are oblivious to the high-level semantic content of the underlying video. In contrast, this paper deals with high-level video personalization defined as the computation and display of a high-level video summary that maximizes the semantic relevance of the personalized content to the client-initiated query while simultaneously satisfying the client-side and system-wide resource constraints and minimizing the client-experienced latency [13]. Constraints imposed by the client-side and server-side security and privacy requirements and by the geospatial distribution and dynamic trajectories of the mobile clients relative to the server(s) are not considered in the current paper; this is a topic for future research.

Proxy-based caching has proved to be a reliable strategy for improving the performance and scalability of distributed systems for multimedia content delivery since the mobile clients may reside in network locations that are remote from the origin servers. Caching not only reduces the client-experienced latency but also alleviates the load on the origin servers. However, video personalization may undermine the benefits of caching since personalization naturally limits the opportunities for content reuse. This research presents the design and implementation of a novel collaborative caching-based video content distribution framework that is *personalization-aware*. The proposed video delivery framework is characterized by the following unique features:
(a) The caches incorporate a multi-stage client request aggregation scheme to reduce the time taken by the caching proxies to generate and disseminate the personalized video and consequently reduce the client-experienced latency [13].
(b) The framework incorporates a novel cache replacement policy that is aware of the existence of different personalized versions of the same underlying video data [12].
(c) The framework incorporates a novel latency-biased collaborative caching protocol based on counting Bloom filters to support the efficient caching of personalized video content and to further enhance cache effectiveness.
(d) The framework supports multiple video personalization servers (VPS's) and proxy caches, with varying degrees of content replication amongst the VPS's.

We study the effectiveness and efficiency of the proposed framework for dissemination of personalized video in a networked mobile environment via extensive experimental evaluation. The experimental results show that the techniques and mechanisms underlying the proposed framework yield substantial performance benefits.

2. SYSTEM OVERVIEW

The proposed framework consists of one or more video personalization servers (VPS's) [4] and several geographically-distributed caches at the edge of network [10]. The VPS's perform automatic video segmentation and index the segmented videos based on the semantics of their contents [14]. The personalized video stream is generated by obtaining a solution to the Multiple Choice Multidimensional Knapsack Problem (MMKP) [14]. The MMKP has also been used in peer-to-peer (P2P) systems for optimal dissemination of layered video content [6]. The caches serve the client requests by performing on-the-fly composition of personalized video streams based on the clients' content preferences and the specified client-side and system-wide resource constraints [4]. The proxy caches are transparent to the clients in the sense that the client requests are automatically redirected to the best proxy based on criteria such as proximity and load. However, our system currently assumes that a client completes its video download before it moves to a different geographic location. Upon receiving a client request, the proxy cache determines whether or not the video segments needed for serving the request are locally available. If so, the proxy cache generates a personalized video stream from the locally available video segments and transmits it to the requesting client(s); if not, the proxy cache first obtains the relevant video segments from other nearby caches or from the VPS's before generating the personalized video stream and transmitting it to the requesting client(s). The proposed framework can be incorporated within an existing content delivery network (CDN). While many CDNs use some form of collaborative caching [2], the proposed framework is unique in that the proxy caches are actively engaged in high-level personalization of video content. By enabling more efficient caching and delivery of layered video, the proposed framework can potentially improve the performance of existing CDNs that host large amounts layered video content.

In the proposed system, the communication between clients and the proxy cache typically occurs in two phases. In the first phase, a client initiates a request for the personalized video content to be generated whereas in the second phase, the client actually downloads the generated personalized content. The proxy cache gathers important statistics about the client requests in order to personalize the video to be disseminated in the second phase of communication. The proposed framework supports multiple levels of client privilege in terms of the visual quality and content abstraction of the delivered video. Currently, the proposed framework supports two levels of client privilege, termed as *subscribing* and *non-subscribing* with the former having priority over the latter. The proxy cache design uses a content-aware video encoding scheme that entails *Features-, Motion- and Object-Enhanced Multi-Resolution* (FMOE-MR) encoding of the texture within each video frame [5] combined with a *Generative Sketch* (GS) encoding of the object outlines in each video frame [3]. By tuning the parameters of the FMOE-MR and GS encodings, it is possible to generate multiple versions of the original video with varying degrees of visual quality and content abstraction, and correspondingly varying file sizes [5]. The proxy caches serve multiple clients with varying levels of privilege and resource constraints by generating video streams at different levels of visual quality, thus improving caching performance. Currently, the video streams are delivered at three levels, the original video V_{org}, which is deemed to be of the highest visual quality (and has the largest file size), the FMOE-MR-encoded and GS-encoded video V_{mid}, which is of intermediate visual quality (and has an intermediate file size), and a base-level video V_{base} which is of lowest visual quality (and has the smallest file size) based on the client-side and system-wide resource constraints and the two levels of client privilege.

The videos hosted by the VPS are segmented and indexed based on their content and then transcoded at multiple levels of content abstraction [14]. A stochastic hierarchical

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* or *semantic concepts* [14]. A semantic unit is a video segment within a video stream 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. The semantic units within a video stream are spliced together to form a logical video sequence that the viewer can comprehend and appreciate. An HMM is formulated for each individual semantic unit. The optimal HMM parameters for each semantic unit are learned from the feature vector sequences obtained from the training video data. The HMMs for individual semantic units are trained separately using the training feature vector sequences, thus allowing for modularity of content [14].

3. PROXY CACHE DESIGN

The proxy cache receiving a client request assumes responsibility for composing the corresponding personalized video stream and delivering it to the client. First, the proxy cache decides which video segments should be included in the video stream by computing the relevance of the individual video segments to the client request. Second, it obtains the segments from other caches or from the VPS's (if they are not locally available) in order to compose the video stream. Third, the proxy cache replaces some of the video segments in its local storage to make space for the incoming video segments. Finally, the video stream is personalized in order to satisfy the client-side and system-wide resource constraints. The first and the final steps are jointly referred to as *personalized video stream composition* and modeled as determining a solution to the MMKP [14].

The video segments, resulting from the segmentation of the video stream, are indexed using semantic terms derived from the *WordNet* ontology [8]. The proxy cache assigns a relevance value V_i to the video segment S_i by computing the similarity between the semantic term T_i used to index the video segment and descriptive term P specified in the client's request. The similarity function is evaluated using the *lch* algorithm [11] which computes the tree traversal distance between the terms T_i and P in the *WordNet* hierarchy. Each indexed video segment is summarized at multiple levels of abstraction using a set of key frames and motion panoramas [14]. For each video segment, its original version is assumed to contain the highest amount of detail; whereas its summary at the highest level of abstraction is assumed to contain the least amount of detail. The Zipf function [15] is used to model the information content of a video summary relative to its original version given the relative duration of the video summary. An MMKP-based video personalization strategy is used to generate a personalized response that maximizes the semantic relevance to the client's request while satisfying multiple client-side and system-level resource constraints [14]. Currently, the MMKP is solved using a Branch-and-Bound Integer Programming (BBIP) algorithm implemented in MATLAB [9].

Since a proxy cache typically receives simultaneous requests from several clients, composing a personalized response to each individual client request places significant load on the proxy cache and on the VPS's. Multi-stage aggregation of several client requests into a single request based on arrival time, video content preference and the client-side resource constraints is employed to alleviate the computational load on the proxy caches and VPS's, and to also reduce the network bandwidth consumption and the client-experienced latency [13]. The multistage client request aggregation is shown to result in significant resource savings in terms of client battery power, network bandwidth and server CPU cycles while delivering personalized video content that most closely matches the clients' request(s) with minimal client-experienced latency [13].

The proposed cache replacement policy is novel in that it is cognizant of the potential existence of multiple personalized versions of a video segment at varying levels of content abstraction, i.e., V_{org}, V_{mid} and V_{base}, where each version represents a distinct level of visual quality with a corresponding file size. The proxy caches are capable of generating V_{mid} and V_{base} from V_{org} in real time. The personalization-aware cache replacement policy is also cognizant of the client's level of privilege, i.e., *subscribing* or *non-subscribing*. The subscribing clients pay for and are guaranteed to receive the best quality video V_{org} whereas for non-subscribing clients the proxy cache provides the best quality video from amongst the versions V_{org}, V_{mid} and V_{base} present in the cache. If none of the versions V_{org}, V_{mid} and V_{base} is present in the proxy cache, the client request is treated as a cache miss in the case of a non-subscribing client. In contrast, in the case of a subscribing client, the client request is treated as a cache miss if the version V_{org} is absent from the cache.

In the event of a cache miss, regardless of the client's level of privilege, the requested file is relayed from the VPS, and stored in the proxy cache simultaneously. If required, the cache replacement policy is enforced to replace (an) existing file(s) in the cache in ascending order of retention value RV (i.e., lowest RV value first) to make space for the requested file. For the proposed *Number of Common Clients over Size* (NCCS) cache replacement policy, we compute the RV as follows: $RV = \frac{NCC}{FS}$, where NCC is the number of common clients requesting that particular file and FS is the file size [12]. The video file with the minimum RV is first identified and the versions V_{org}, V_{mid} and V_{base} of the identified video file removed in that order until there is enough space to load the requested file in the proxy cache. This ensures that the version of a video segment which takes up the largest amount of space in the cache is removed first, so that space for more popular video segments can be made. The proxy cache generates V_{mid} and V_{base} from V_{org} in real time after fetching V_{org} from the VPS and stores all of them locally. The total storage requirement of all the files V_{org}, V_{mid} and V_{base} is approximately 1.5 times that of V_{org} resulting in an $\approx 50\%$ storage overhead for the video segment. By ensuring a reasonably good visual quality for V_{mid} and V_{base}, it possible to discard V_{org} to allow for extra space in the cache, and yet have videos of reasonable visual quality resident in the cache, during cache replacement. The proposed NCCS cache replacement policy is shown to outperform other commonly used cache replacement policies such as the Least Recently Used (LRU) and Least Frequently Used (LFU) cache replacement policies [12].

4. COLLABORATIVE CACHING

The proposed framework supports collaboration amongst the proxy caches to further improve scalability and efficiency of personalized video delivery. Collaboration amongst proxy caches that serve a common client pool greatly reduces the number of server accesses that need to be performed by the proxy caches thereby improving the scalability of the system. Moreover, since accessing content from a nearby proxy cache is usually faster and cheaper in terms of bandwidth consumption on the network backbone, collaborative caching reduces significantly the client-experienced latency [10]. In a collaborative caching scheme, in the event of a cache miss, the proxy cache tries to locate the corresponding video segment in nearby proxy caches. Only if none of the nearby proxy caches has the video segment, the closest VPS is contacted. To account for varying levels of client privilege, if the requesting client is subscribing, we obtain the V_{org} version of the requested video, even if it requires accessing the VPS. If the client is non-subscribing, we access the best version that is available on the nearby proxy cache in order to serve the non-subscribing client.

In the proposed collaborative caching scheme, each proxy cache maintains up-to-date information about the contents of other nearby caches using *counting Bloom filters* [1], [7]. A counting Bloom filter is a collection of n-bit counter arrays which are mapped using a number of hash functions in order to support membership queries. Several hash functions are used to map elements with distinct ID's to their corresponding n-bit counters in the counting Bloom filter. For querying whether or not an element is present, the element ID is fed to all the hash functions, and it is verified whether or not the n-bit counter output of each hash function has a value greater than zero; if so, the element is deemed to be present else absent. Initially, all the bits in each counter are set to zero. In the event that a video segment is added to a cache, the n-bit counter corresponding to each of the hash functions is incremented by one. Likewise if a video segment is deleted, the n-bit counter corresponding to each of the hash functions is decremented by one.

In the proposed collaborative caching scheme, each proxy cache constructs and maintains a counting Bloom filter which, at all times, accurately reflects the status of the video segments that are currently resident in the proxy cache. The concatenation of the video segment ID and the version ID is used as the index for the counting Bloom filters. The collaborating proxy caches also exchange their counting Bloom filters periodically. Thus each proxy cache has copies of the counting Bloom filters of all the proximal proxy caches. However, these counting Bloom filter copies available at an individual proxy cache do not reflect the additions and deletions that have occurred at the corresponding proxy caches since the last exchange. When a proxy cache needs a particular video segment to serve a client request, it first checks whether a locally available video segment can be used for serving the user request. If the client is a subscribing client, a video segment is considered to be usable if it not only has the requested ID but is also the V_{org} version of the video requested by the client. If the client is a non-subscribing client, a video segment with matching ID regardless of its version is considered to be usable. If no usable video segment is available locally, the proxy cache checks whether a usable video segment is available in proximal proxy caches using the corresponding counting Bloom filter copies. If so, it contacts the proxy cache where the segment is available, obtains it, and serves the client. If none of the nearby proxy caches have the video segment, the request is sent to the origin VPS. Notice that both false hits and false misses are possible although their probabilities can be made very small by more frequent exchanges between the counting Bloom filters. False hits are resolved at the proxy cache serving the client request by sending a request for the relevant video segment to the origin VPS.

5. EXPERIMENTAL EVALUATION

An experimental prototype of the proposed personalized video content delivery framework was implemented and evaluated. The experimental evaluation had three goals: (a) evaluating the impact of cache cooperation on the scalability and performance of the system, (b) evaluating the performance of the system using multiple VPS's with varying degrees of mirrored content, and (c) studying the limitations of cache cooperation in a ring-based network topology. The VPS's and proxy caches were deployed on 2.79GHz Pentium-4 workstations, each with 2GB of RAM and running the Linux RedHat-4 operating system. The number of clients per VPS was 150 with 40% of the clients being subscribing and the remainder non-subscribing.

The client preference value was generated randomly using a uniform distribution over the elements of the specified client preference vector. The resource constraints, namely, the viewing time limit and the data bandwidth, were each modeled using a normal distribution. In order to simulate mobile devices with different viewing time limits, the values for the viewing time limit were modeled as a mixture of 2 normal distributions $\mathcal{N}_1(\mu_1, \sigma_1)$ and $\mathcal{N}_2(\mu_2, \sigma_2)$, where μ_1 = 50 seconds, σ_1 = 70 seconds and μ_2 = 150 seconds, σ_2 = 70 seconds. The data bandwidth was modeled as a normal distribution $\mathcal{N}_3(\mu_3, \sigma_3)$ where μ_3 = 50 Kbps and σ_3 = 40 Kbps. Our simulations involved multiple sessions of client-cache-server communication; each session consisting of multiple client requests which followed a Poisson distribution with a maximum delay of 4 seconds and variance of 2 seconds.

The proxy cache performance was evaluated using standard metrics such as the *hit ratio*, *byte hit ratio* and *client-experienced latency*. We made a distinction between a *local* cache hit where the client request is served by a proxy cache using locally available content and a *remote* cache hit where the requested content is obtained from another proxy cache. If the content is obtained from the VPS then the client request was termed a *miss*. The *local (remote) hit ratio* was defined as the number of local (remote) cache hits as a percentage of total number of client requests. The *cumulative hit ratio* was defined as the sum of the local and remote cache hit ratios. The various byte hit ratios were defined analogously. The *client-experienced latency* was computed as the elapsed time between the instant that the client submits a request during the first phase of communication and the instant it receives the first byte of the first requested video segment during the second phase of communication.

We compared the performance of proposed NCCS cache replacement policy with that of the more commonly used LFU cache replacement strategy using the parameters *remote hit ratio* and *remote byte hit ratio*. For the LFU cache replacement policy the retention value is given by $RV_{LFU} = N_h$ where N_h is the number of hits to the video segment since it was cached. We performed various experiments to compare the effectiveness of collaboration between the proxy caches to the scenario where there is no collaboration amongst the proxy caches for both, the NCCS and LFU

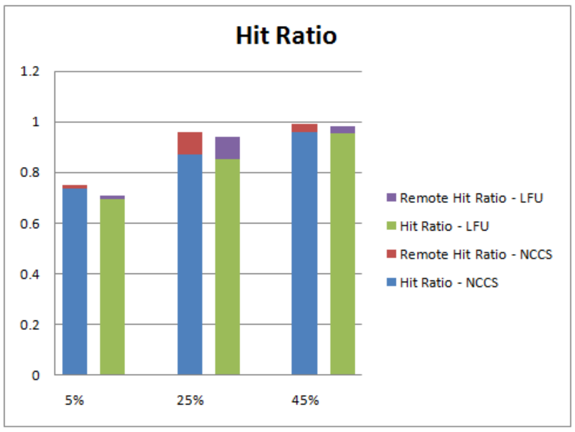

Figure 1: Cache hit ratio for NCCS and LFU with varying cache sizes

cache replacement policies for varying cache sizes (measured as a percentage of VPS storage capacity). As shown in Figure 1, the NCCS cache replacement policy was observed to outperform the LFU cache replacement policy for varying cache storage capacities, both with and without collaboration.

Figure 2 shows the mean client-experienced latency (computed as a percentage of the client-experienced latency in the absence of caching) for the NCCS and LFU cache replacement policies with and without inter-cache collaboration for varying cache sizes from 5% to 95% of the VPS storage capacity. It was observed that the collaborative versions of the NCCS and LFU cache replacement policies exhibit a lower client-experienced latency than their non-collaborative counterparts. Also, the NCCS cache replacement policy, with and without inter-cache collaboration, was observed to yield a lower client-experienced latency than its LFU counterpart. The client-experienced latency was observed to be a decreasing function of cache size in all cases. In the case of the NCCS cache replacement policy, for a cache size of 45% of the VPS storage capacity, inter-cache collaboration was observed to result in a $\approx$ 20% reduction in the client-experienced latency. Figure 3 depicts the performance of the counting Bloom filter, in terms of the client-experienced latency, as a function of the Bloom filter size (measured in terms of the number of bits allocated per file). It was observed that the caching performance improves as the counting Bloom filter size increases since a larger counting Bloom filter size provides greater discrimination between distinct files. However, the performance improvement was observed to be more significant when the Bloom filter size is increased from 8 bits to 16 bits than when it is increased from 16 bits to 32 bits, implying diminishing gain in performance as the Bloom filter size is increased.

The limitations of cache collaboration were examined. The proxy caches were assumed to be connected in a network with a ring topology. In the event of a miss, a proxy cache first contacts its nearest neighboring proxy cache(s) in the network (at a single-hop distance), moving on to successively farther proxy caches or VPS's in the network until the requested content is obtained. In these experiments, we examined three distinct scenarios, full collaboration, latency-biased collaboration and no collaboration amongst the proxy

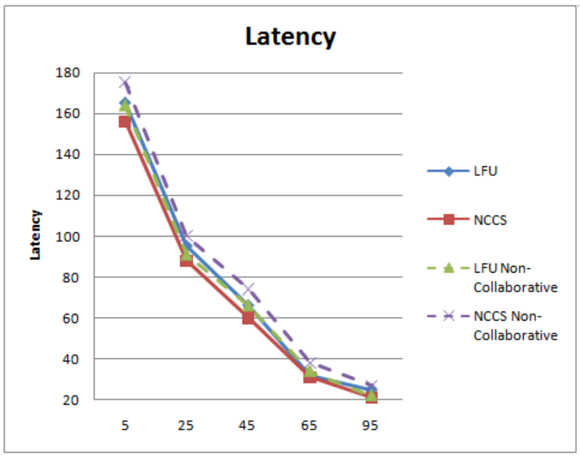

Figure 2: The average client-experienced latency for collaborative and non-collaborative caching as a function of cache size with NCCS and LFU.

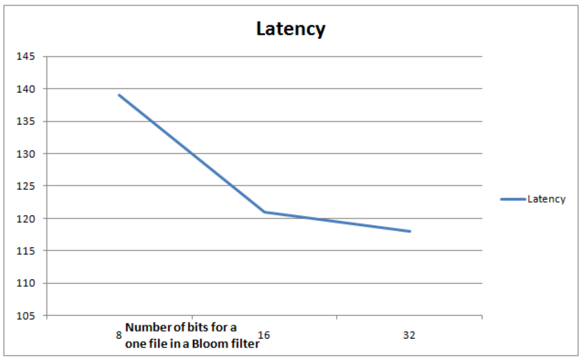

Figure 3: Mean client-experienced latency as a function of the counting Bloom filter size.

caches. In the case of full collaboration amongst the proxy caches, a proxy cache, in the event of a miss, contacts all the proxy caches in the network for the needed content, before contacting any of the VPS's. In latency-biased collaboration, a proxy cache, upon encountering a miss, determines its nearest neighbor in the network, be it a proxy cache or a VPS. If a proxy cache determines that a VPS is nearer than another proxy cache in the network, it contacts the VPS instead of the proxy cache for the needed content. In the complete absence of inter-cache collaboration, a proxy cache, upon encountering a miss, contacts the nearest VPS for the needed content. Figure 4 shows the client-experienced latency for varying cache sizes for all the three aforementioned scenarios. As expected, the latency-biased collaborative caching scheme was observed to exhibit the lowest client-experienced latency whereas the scheme with no inter-cache collaboration was observed to exhibit the highest client-experienced latency. The performance of the fully collaborative caching scheme was seen to lie between that of the latency-biased collaborative caching scheme and the scheme with no inter-cache collaboration. In the case of all the three aforementioned schemes, the client-experienced latency was observed to be a decreasing function of proxy cache size. We varied the replication of content in the various VPS's

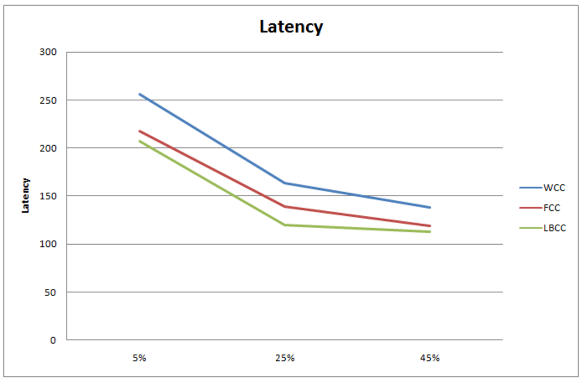

Figure 4: The average client-experienced latency for three scenarios with varying proxy cache sizes; FCC: Fully Collaborative Caching; WCC: Without Collaborative Caching; LBCC: Latency-biased Collaborative Caching.

from 0% (i.e., no replication) to 100% (i.e., fully replicated or mirrored content). The mean client-experienced latency was observed to decrease with increase in the extent of content replication amongst the VPS's. We also examined the mean client-experienced latency as a function of the maximum network hop distance over which a proxy cache can collaborate with neighboring proxy caches (i.e., sphere of inter-cache collaboration). The mean client-experienced latency was observed to increase with increase in the sphere of inter-cache collaboration implying thereby that the proxy cache is better off contacting the nearest VPS than a more distant proxy cache in the network.

6. CONCLUSIONS & FUTURE WORK

We proposed a collaborative caching mechanism for personalized video content delivery consisting of one or more VPS's and several geographically distributed proxy caches. The VPS's used an automatic video segmentation and video indexing scheme based on semantic video content and employed an MMKP-based video personalization strategy that accounted for client preferences and resource constraints. The proxy caches incorporated a novel multi-stage client request aggregation algorithm, a unique personalization-aware cache replacement policy and a counting Bloom filter-based mechanism to collaboratively serve client requests. A latency-biased collaborative caching (LBCC) scheme was proposed which dynamically adjusts the sphere of inter-cache collaboration, based on the inter-cache latency and the cache-VPS latency values. The proposed LBCC scheme was shown to be superior to the fully collaborative and non-collaborative caching schemes. Future work will deal with mobile clients with privacy constraints and constraints arising from geospatial locations and geospatial trajectories of the mobile clients in addition to resource constraints. The current framework will be enhanced to enable interactive querying of streaming video data in order to support interactive video stream control functions such as skip, rewind and fast-forward. We will also explore video personalization and video delivery in ad-hoc peer-to-peer networks characterized by resource, geospatial and privacy constraints and integrating the proposed FMOE-MR video encoding scheme with the current MPEG-4 SVC standard.

7. REFERENCES

[1] Bonomi, F., Mitzenmacher, M., Panigrahy, R., Singh, S., and Varghese, G. (2006) An Improved Construction for Counting Bloom Filters, Y. Azar and T. Erlebach (Eds.): *Proc. Eur. Symp. Algorithms* (ESA 2006), Zurich, Switzerland, Springer LNCS 4168, pp. 684-695.

[2] Buyya, R.C., Pathan, M., and Vakali, A. (Eds.) (2008) *Content Delivery Networks*, Springer, New York, NY.

[3] Chattopadhyay, S., Bhandarkar, S.M. and Li, K. (2007) Ligne-Claire Video Encoding for Power Constrained Mobile Environments, *Proc. ACM Multimedia*, Augsburg, Germany, Sept. 24-29, pp. 1036-1045.

[4] Chattopadhyay, S., Ramaswamy, L., and Bhandarkar, S.M. (2007) A Framework for Encoding and Caching of Video for Quality Adaptive Progressive Download, *Proc. ACM Multimedia*, Augsburg, Germany, Sept. 24-29, pp. 775-778.

[5] Chattopadhyay, S., and Bhandarkar, S.M. (2008) Hybrid Layered Video Encoding and Caching for Resource Constrained Environments, *Jour. Visual Comm. Image Rep.*, Vol. 19(8), Dec., pp. 573-588.

[6] Eberhard, M., Szkaliczki, T., Hellwagner, H., Szobonya, L., and Timmerer, C. (2010) Knapsack Problem-based Piece-picking Algorithms for Layered Content in Peer-to-Peer Networks, *Proc. ACM Workshop AVSTP2P*, Oct. 29, Florence, Italy.

[7] Fan, L., Cao, P., Almeida, J. and Broder, A.Z. (2000) Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol, *IEEE/ACM Trans. Networking*, Vol. 8(3), pp. 281-293.

[8] Fellbaum, C. (Ed.) (1998) *WordNet - An Electronic Lexical Database*, The MIT Press, Cambridge, MA.

[9] Hernandez, R.P., and Nikitas, N.J. (2005) A New Heuristic for Solving the Multichoice Multidimensional Knapsack Problem, *IEEE Trans. SMC*, Part A, Vol. 35(5), pp. 708-717.

[10] Jin S., and Bestavros, A. (2000) Popularity-aware Greedy Dual-size Web Proxy Caching Algorithms, *Proc. 20th IEEE ICDCS*, Taipei, Taiwan, April 10-13, pp. 254-261.

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

[12] Parate, P., Ramaswamy, L., Bhandarkar, S.M., Chattopadhyay, S., and Devulapally, H.K. (2009) Efficient Dissemination of Personalized Video Content for Mobile Environments, *Proc. IEEE Intl. Conf. CollaborateCom*, Washington, DC, Nov. 11-14.

[13] Wei, Y., Bhandarkar, S.M. and Li, K. (2007) Video Personalization in Resource-Constrained Multimedia Environments, *Proc. ACM Multimedia*, Augsburg, Germany, Sept. 24-29, pp. 902-911.

[14] Wei, Y., Bhandarkar, S.M. and Li, K. (2009) Client-centered Multimedia Content Adaptation, *ACM TOMCCAP*, Vol. 5(3), pp. 22:1 - 22:26.

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