Caching Strategies in Transcoding-enabled Proxy Systems for Streaming Media Distribution Networks

Bo Shen, Sung-Ju Lee, and Sujoy Basu

***Abstract*— With the wide availability of high-speed network access, we are experiencing high quality streaming media delivery over the Internet. The emergence of ubiquitous computing enables mobile users to access the Internet with their laptops, PDAs, or even cell phones. When nomadic users connect to the network via wireless links or phone lines, high quality video transfer can be problematic due to long delay or size mismatch between the application display and the screen. Our proposed solution to this problem is to enable network proxies with the transcoding capability, and hence provide different, appropriate video quality to different network environment. The proxies in our transcoding-enabled caching (TeC) system perform transcoding as well as caching for efficient rich media delivery to heterogeneous network users. This design choice allows us to perform content adaptation at the network edges. We propose three different TeC caching strategies. We describe each algorithm and discuss its merits and shortcomings. We also study how the user access pattern affects the performance of TeC caching algorithms and compare them with other approaches. We evaluate TeC performance by conducting two types of simulation. Our first experiment uses synthesized traces while the other uses real traces derived from an enterprise media server logs. The results indicate that compared with the traditional network caches, with marginal transcoding load, TeC improves the cache effectiveness, decreases the user-perceived latency, and reduces the traffic between the proxy and the content origin server.**

***Index Terms*— proxy caching, streaming media, video transcoding, network measurements, streaming in wireless networks, streaming media distribution**

I. Introduction

THE Internet is growing everyday; we see more sites providing more content to more users each day. For efficient content delivery, proxy caches are deployed at the network edges [1]. Popular web objects are cached at a proxy near the users so that the traffic between the content origins and proxies reduces and the user perceived latency decreases. Streaming media distribution over the Internet is a different challenge. Encoding, delivery, caching, and processing are far more difficult than simple web objects such as HTML and image files. Larger object size, static content property, and different user access behaviors require a different caching system than traditional web caches. Several schemes have been proposed for streaming media caching in the proxy. Most of these proposals however, implicitly assume that the end users have similar network environment and computing capability.

Different Internet users have heterogeneous network and computing environments. There are those who have high-speed network access through corporate LANs, DSL, or cable modems. Some people dial up using a modem to connect to the Internet, and there are mobile users who browse Internet content on their laptop, PDAs, and cellular phones using low-bandwidth wireless links. Users in fast networks prefer high resolution videos while users without high-speed network access may not enjoy high quality videos because the delay is large and the video may not fit within the device display. Many websites encode streaming video clips at several different bit-rates (28$\sim$56 kbps for dial-up connections and 150-plus kbps for broadband networks) to satisfy users with different network connection speeds. Existing media caching systems treat each client request equally and independently. Various different bit-rate versions of the same video clip may be cached at the proxy at the same time, which is a waste of storage.

We introduce Transcoding-enabled Caching (TeC) proxies for streaming media distribution over the Internet. Our system is designed for efficient delivery of rich media web content to heterogeneous network environments and client capabilities. The proxies in our system perform transcoding as well as caching. Depending on the connection speed and processing capability of an end user, the proxy transcodes the requested (and possibly cached) video into an appropriate format and delivers it to the user. By putting the transcoding unit on the content delivery path, we perform content adaptation at the network edges. One potential advantage of TeC is that the content origin servers need not generate different bit-rate versions (although we assume in our experiments that multiple versions are available). Moreover, heterogeneous clients with various network conditions will receive videos that are suited for their capabilities, as content adaptation is more appropriately done at the network edges.

There are many coding techniques that are developed to manage heterogeneous clients. Scalable coding and layered coding are the typical examples. These techniques are included in some of the video coding standards such as MPEG-2 and MPEG-4. However, majority if not all of the current multimedia content on the Internet is coded in non-scalable, single-layered format. Given the lack of layered-coded contents, we argue that using transcoding to adapt to the user heterogeneity is a more practical approach.

We propose three caching strategies for TeC. These schemes take into account that variants of the same video may exist in the system. The first two algorithms cache at most one version of a video object. They operate differently when a user requests a video version that is coded at a lower bit-rate than the one cached in the proxy. The third algorithm may cache multiple versions of the same video object and hence reduces the processing load at the transcoder. We describe each algorithm in detail and highlight its merits and shortcomings.

The authors are with Hewlett-Packard Laboratories, Palo Alto, CA.

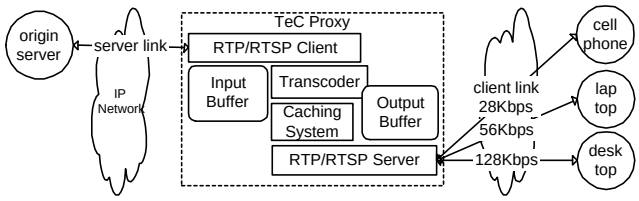

Fig. 1. System and components for the transcoding-enabled caching proxy.

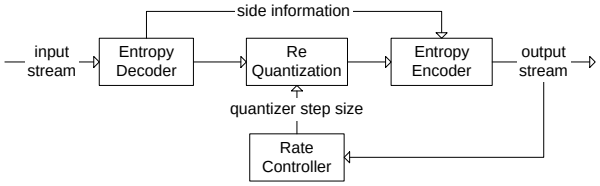

Fig. 2. Bit-rate reduction transcoding.

The rest of the paper is organized as follows. Section II introduces TeC system architecture along with our transcoding technique and caching algorithms. We present performance evaluation in Section III followed by related work in Section IV. We conclude in Section V.

II. Transcoding-enabled Caching (TeC) System

A. System Architecture

Transcoding-enabled caching (TeC) proxy consists of the components shown in Figure 1. The proxy acts as a client to the content server. An RTP/RTSP client is built into the proxy to receive the streamed content from the origin server (server link). The received stream is put into the input buffer. The transcoder continuously pulls bit streams from the input buffer and subsequently pushes the transcoded bits out to the output buffer. The proxy caches the content either from the input buffer or the output buffer while the transcoder produces the content. Additionally, the proxy acts as a server to the end user. Hence, an RTP/RTSP server is built to stream the video to the end user (client link). The data in the output buffer is obtained either from the transcoder or the caching system.

The size of the input and output buffers can be small given that the transcoder processes the video data in a streamlined fashion. The speed of the process, i.e., the transcoding bit-rate, is defined as the number of bits the transcoder generates per second. Given that the transcoding bit-rate is larger than the minimum of the server link and the client link bandwidths, the transcoding process does not significantly increase the end-to-end delay.

Given the real time transcoding capability, the TeC proxies dynamically transcode video objects to different variants to satisfy the end users in heterogeneous environments. Each variant is a version. If version x can be obtained from transcoding version y, we call version y a *transcodable version* for x. Conversely, version x is the *transcoded version* of y. In video transcoding, a higher bit-rate version can be transcoded to a lower bit-rate version. For example, if a video at bit-rate of 64 kbps can be transcoded from the same video at bit-rate of 128 kbps, the 128 kbps version is a transcodable version for the one at 64 kbps. Consequently, the 64 kbps version is a transcoded version from the one at 128 kbps.

The transcoded version may have fidelity degradation compared with the original version. The TeC proxy can produce transcoded versions with 1 to $(n-1)$ generation loss in fidelity, where n is the total number of possible versions. For video transcoding, this loss is negligible when bit-rate reduction is coupled with resolution reduction. For example, when a video clip with the CIF resolution (352×288) at bit-rate of 1 Mbps is to be delivered to a PDA type of client device with resolution at QCIF (176×144), the reduction in the spatial resolution already yields the bit-rate reduction by a factor of four.

Note that the variation in versions delivered to the client may not be transparent to the end user. The end user specifically asks for a certain version of an object based on the awareness of ones connection and display device. Alternatively, a client agent software informs the client's connection and device capability to the proxy. The proxy then chooses a version for the session. The TeC proxy is often located at the edge of the network close to the end user. If there are packet losses in the server link due to congestion, the TeC proxy can choose not to cache for the session.

B. Transcoding Techniques

Video transcoding is a computation-intensive task. Many researchers have developed efficient methods to reduce the workload of a transcoding session [2]. Among those, compressed domain based approach provides the best performance. In compressed domain transcoding, the input video is only partially decompressed. Rate adaptation is performed in the compressed domain while the motion information is reused. This approach considerably improves the processing speed over the conventional decode-and-re-encode approach.

The TeC proxy utilizes compressed domain transcoding techniques. We introduce two types of transcoder, namely bit-rate reduction and spatial resolution reduction transcoding. Figure 2 illustrates a bit-rate reduction transcoding process. DCT coefficients are obtained through entropy decoding. They are subsequently re-quantized using a coarser quantization step size produced by the rate-control module. All other side information of the bit stream is directly transmitted to the entropy encoder, or directly to the output bit stream if no modifications are required. Since no pixel-domain frames are reconstructed, the transcoding process is significantly faster than a decode-and-re-encode approach. We model the TeC proxy based on this transcoder later in our performance evaluation section. [1]

Spatial resolution reduction transcoding can also be implemented in the compressed domain. The transcoder obtains motion information for the down-sampled video directly from the original video and therefore eliminates a costly motion estimation process. Readers are referred to [3] for system design and performance evaluation.

Both of these two methods utilize the compressed domain information as much as possible. Hence the transcoding process is conducted with significantly less CPU load. This

[1] We show the performance of the transcoder itself in the Appendix.

characteristic is important for the transcoding-enabled proxy since there may be intensive transcoding. We assume sufficient computing power at the TeC proxy and focus primarily on investigating the caching benefits enabled by the collocated transcoder.

C. Transcoding-enabled Caching Algorithms

With the proxy being able to transcode and cache the video object, the main goal of TeC is to serve the end user with the appropriately transcoded version of the cached video whenever possible, based on the network capacity and connection profile of the user. We propose to trade off computation with storage.

We define the following events in a TeC proxy:

- *Exact hit*: the requested version of the video object exists in the cache.
- *Transcode hit*: the requested version does not exist in the cache, but a transcodable version of the video exists.
- *Miss*: the requested or a transcodable version of the video does not exist in the cache.

Let us assume that the origin server has n versions at bit-rates $b_0, b_1, \ldots, b_{n-1}$ for each video object. The highest bit-rate version is b_0 and the lowest is b_{n-1}, i.e., $b_0 > b_1 > \ldots > b_{n-1}$. When version b_j is requested from the end user and there is version b_i ($b_i > b_j$, i.e., b_i is a transcodable version for b_j) in the cache, the TeC proxy transcodes b_i to b_j and sends the transcoded b_j to the client instead of fetching b_j from the content origin. Therefore, it is a cache hit (i.e., a *transcode hit*) even though b_j is not directly available from the cache at the time of request.

Note that when the origin server has only one bit-rate version of a video (possibly a high bit-rate) and a user with low-speed connectivity requests that object, our proxy transcodes the original video into an appropriate bit-rate object and streams it to the user. Hence, TeC works well regardless of whether the content origin supports various bit-rate versions of the videos or not.

We propose three different caching algorithms for the TeC system. TEC-11 and TEC-12 cache at most one version of a video object at the proxy at any time. These two algorithms operate differently when there is a *transcode hit*. On the other hand, TEC-2 may cache multiple versions of the same video and hence reduces the processing load on the transcoder.

1) Cache Single Version (TEC-11 and TEC-12): This algorithm allows at most one version of a video object to be cached at the proxy at any single time. By caching only one version, we store more video objects and efficiently utilize the storage space. The main challenge of this algorithm is deciding which bit-rate version of the video to cache.

When an exact cache hit occurs, the TeC proxy refreshes the access record of the object and streams it to the client. If a request leads to a cache miss, the TeC proxy fetches the video from the origin server, transcodes it if needed, streams it to the user and caches it. Remember that we consider each version of a video as an independent item; although a request is to a video that is cached, if the request had to be responded from the content origin (i.e., the user requests a higher bit-rate version than the cached one), then it is a cache miss.

```
Version b_i of a video object is requested...

if b_i is already in the proxy cache
   stream b_i to the user from the cache
   update the access record of b_i
else if b_j of the same video is in the cache and b_j > b_i
   transcode b_j into b_i
   stream the transcoded b_i to the end user
   if TEC-11
      update the access record of b_j
   if TEC-12
      evict b_j from the cache
      store b_i in the cache
      update the access record of b_i
else if b_j is in the cache and b_j < b_i
   evict b_j from the cache
   fetch version b_i of the video from the origin
   stream b_i to the end user
   if the cache space is not enough for b_i
      evict victims using a replacement algorithm
   store b_i in the cache
   update the access record of b_i
else
   fetch version b_i of the video from the origin
   stream b_i to the end user
   if the cache space is not enough for b_i
      evict victims using a replacement algorithm
   store b_i in the cache
   update the access record of b_i
```

Fig. 3. Cache single version, algorithms TEC-11 and TEC-12.

When a user requests version b_i of a video while b_j, where $b_i > b_j$, exists in the cache, b_j is removed before b_i is fetched from origin server and subsequently cached at the proxy. Since we allow only one version of an object to be cached, the lower bit-rate version is evicted from the cache. If a request results in a transcode hit, the TeC proxy transcodes the cached object to an appropriate bit-rate and streams it to the user. In the mean time, the proxy chooses which version to cache in two different ways, which leads to two variations of the algorithm. For algorithm TEC-11, the proxy refreshes the access record of the already cached object (i.e., the higher bit-rate) without caching the newly transcoded version. For algorithm TEC-12, the proxy evicts the transcodable version from the cache and stores the newly transcoded version. In summary, if the client requests version b_j of a video while b_i, where $b_i > b_j$, exists in the cache, b_j is transcoded from b_i and streamed to the user. TEC-11 refreshes the access record of b_i, but TEC-12 removes b_i and caches b_j.

Whenever the cache is full and requests to the un-cached video are received, certain files in the cache must be replaced. We use the existing popular cache replacement algorithms (e.g., LRU, LFU, LRU-k [4], or GD* [5]) for this purpose. The pseudocode of TEC-11 and TEC-12 is presented in Figure 3.

2) Cache Multiple Versions (TEC-2): The motivation of caching multiple versions of the same video object is to reduce the processing load on the transcoder. For example, if b_i and b_j are both in the cache, a request to b_j will lead to an exact hit and no transcoding is needed. In addition, if the accesses to a certain video object across its variants shows high temporal

```
Version b_i of a video object is requested...

if b_i is already in the proxy cache
   stream b_i to the user from the cache
   update the access record of b_i
else if b_j is in the proxy cache and b_j > b_i
   transcode b_j into b_i
   stream the transcoded b_i to the end user
   if the cache space is not enough for b_i
      evict victims using a replacement algorithm
   store b_i in the cache
   update the access record of b_i and b_j
else
   fetch version b_i of the video from the origin
   stream b_i to the end user
   if the cache space is not enough for b_i
      evict victims using a replacement algorithm
   store b_i in the cache
   update the access record of b_i
```

Fig. 4. Cache multiple versions, algorithm TEC-2.

locality, TEC-2 may further improve the caching efficiency.

When there is a cache miss, the TeC proxy fetches the video from the origin, transcodes it to the requested version if required, streams it to the client and caches it even when other bit-rate versions of the same video object are in the cache. Consequently, multiple versions of a popular video can be cached at a given time. If a transcode hit occurs, the transcoder generates the requested version. It is subsequently delivered to the end user and cached in the proxy. For example, if the client requests version b_j of a video while b_i ($b_i > b_j$) exists in the cache, b_j is transcoded from b_i, delivered to the user, and cached. Note that both b_i and b_j now exist in the cache.

Similar to TEC-11 and TEC-12, when we need space to cache new objects, we use an existing cache replacement algorithm. Figure 4 shows the pseudocode of TEC-2 algorithm.

3) Discussion: The effectiveness of the three algorithms highly depends on the user access behavior and network environment. For instance, when the users connected to a proxy have similar network capacities (e.g., corporate employees during work hours that have high-speed network connection), algorithms TEC-11 and TEC-12 will perform better than TEC-2. In fact, if the proxy has the knowledge of which bandwidth is predominant among the links to the users it is connected to, it will know the appropriate bit-rate for that bandwidth and cache only that version of the video. On the other hand, if the users show heterogeneous network connectivity and processing capability and the access behavior shows strong temporal locality, TEC-2 will show superior performance.

III. Performance Analysis

We implemented a simulator for performance analysis. To focus on evaluating the TeC performance, no partial caching is considered. That is, if an object is cached, the entire object is cached. In practice, TeC can work with partial caching schemes such as prefix caching [6] or segment-based caching [7] to provide better performance. We conduct two types of simulation. First, we use synthesized traces to simulate heterogeneous network environments. We then use an enterprise trace to simulate prolonged access of media content in mostly homogeneous network environment. To our knowledge, no other streaming media caching work has evaluated its performance using actual media traces.

We compare the performance of the TeC proxy with the algorithms proposed in [8] that also combine caching with transcoding of streaming objects. Algorithm FVO (Full Version Only) caches only the full, original version the content origin provides and serves requests to lower versions with transcoding. However, the transcoded objects are not cached. On the other hand, TVO (Transcoded Version Only) always caches the transcoded objects, and if a request does not results in an exact hit, the full version is fetched from the origin to produce a transcoded version. In all the simulations, the TEC, FVO, and TVO use LRU as the cache replacement algorithm in combination with their respective caching strategies.

We also simulate a regular caching proxy that serves only as an interception proxy using LRU without any transcoding capability. To compare the performance of regular caching proxy with that of the TeC proxy, we assume that the content origin server provides multiple bit-rate versions for each video object. Note that having multiple versions at the origin server does not impact the evaluation of caching performance for any algorithms under investigation.

A. Synthesized Trace Driven Simulation

To evaluate the TeC performance in heterogeneous network environments, we use our "synthesized" trace as no public media trace of this kind is available. Two sets of simulations are conducted. Both are based on the same content pool created as follows. We create a pool of 500 original video objects with the running time varying from 5 to 15 minutes. Each video object has the full, original version with a bit-rate of 512 kbps and also has 256 kbps, 128 kbps, and 64 kbps versions.

The first simulation set represents a highly heterogeneous environment. In this simulation, the full version is not requested by the clients as we want to create a scenario similar to the one in [8]. For the second set, we vary the degree of heterogeneity and the object popularity distribution to evaluate the algorithms under different situations. The full, original version is accessed in this scenario. We have also conducted simulations in environments where the origin provides videos with only one bit-rate and accesses are predominantly from a certain client link bandwidth. These results are reported in [9].

1) Highly Heterogeneous Environment: In this scenario, the media content is evenly accessed by clients in each bandwidth capacity. Note that the full version is not requested by the clients but could be accessed by the proxy when FVO or TVO are used. The popularity of the video objects follows a Zipf distribution with the skew factor α of 0.47 [10]. The simulation lasts four hours with 1,000 accesses arriving at a random Poisson process. Consequently, the average access inter-arrival time is 14.4 seconds. During the simulation, a total of 10 GB of content is delivered to the clients.

Figure 5 (a) shows the byte hit ratio as a function of relative cache size. Byte hit ratio is defined as the number of bytes

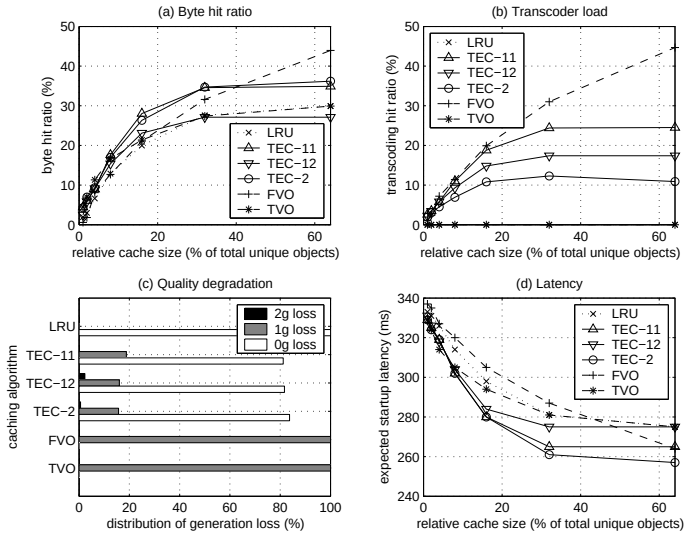

Fig. 5. Performance comparison in model-based simulations.

served from the cache to the clients over the number of bytes requested from the clients. In the case of a transcode hit, the byte of transcoded-down version is used in the calculation. Relative cache size is obtained by dividing the cache size by the size summation of total unique objects (here, different versions of an object are considered unique). TEC-11 and TEC-2 generally provide the highest byte hit ratio. TEC-11 is effective especially when there is limited cache space as it efficiently utilizes the storage. TEC-2 performs well as requests to lower bit-rate versions of cached objects are delivered after transcoding at the proxy, instead of fetching from the content origin. We can observe that TEC-2 outperforms TEC-11 as cache space gets larger. This is an expected result because with large cache, TEC-2 stores multiple popular videos with multiple versions. TEC-12 performance suffers as the eviction of higher bit-rate objects after transcoding at the proxy results in cache misses when clients afterwards request higher bit-rate video versions. In fact, it performs even worse than LRU when the cache size is large. TVO shows similar performance to LRU. FVO does not perform as well as TEC-11 and TEC-2 when the relative cache size is less than 40%. Its performance improves when the relative cache size increases as it stores more full version objects and hence serves requests to lower bit-rate versions by transcoding from the full version. In practice however, large cache may not always be available.

Figure 5 (b) shows the transcode hit ratio. It represents the ratio of the number of cache hits served from transcoding to the total number of user requests. This metric also indicates the transcoding load each algorithm requires. Note that FVO performs the most transcoding, even when it achieves lower byte hit ratio than TEC-11 and TEC-2 (as seen in Figure 5 (a), when cache size is less than 40%). TVO does not have any transcode hit as it does not support transcoding of already transcoded versions and does not cache the full original version. Among TECs, TEC-11 shows the most transcoding load and TEC-2 the least as anticipated. TEC-11 serves nearly 25% of the requests by transcoding the cached objects.

Bit-rate reduction transcoding often introduces quality degradation less than 1 dB [11]. Repeated transcoding introduces further degradation and this is characterized as a generation loss. Figure 5 (c) shows the distribution of generation losses for the objects served from transcoding when the relative cache size is set to 16%. For all TEC algorithms, only 15$\sim$18% of the requests are served with generation losses. There is at most one generation loss for all the requests when TEC-11 is used since it caches the version at higher bit-rate. FVO and TVO do not have more than one generation loss, as they transcode only from the original full version. It is one of the reasons FVO and TVO perform worse than TEC algorithms as FVO and TVO do not support transcoding from transcoded streams that exist in cache. Since the original full version is not accessed in this scenario, FVO and TVO always serve video streams with 1g loss.[2] TEC algorithms also produce video streams with 2g loss. However, as shown in the Appendix, 2g loss is insignificant.

The start-up latency is another important metric in streaming media delivery. We define start-up delay as the time interval between the instance when the proxy receives a request and the instance when the first packet is served to the client. Using the values obtained from Figure 10 (b) of the Appendix, the simulator calculates the start-up delay. Figure 5 (d) shows the results. The start-up delay decreases when cache capacity increases since more requests are served directly from the proxy. Since TEC-11 always caches the higher bit-rate version, requests to the lower bit-rate version of the same object are served with shorter delay as access to the origin server is not needed. However, higher bit-rate version occupies larger space, which reduces the number of exact hits. On the other hand, as the start-up delay for a transcode hit is much shorter than the delay from a miss, the TeC schemes are still favorable in terms of latency. TEC-11 and TEC-2 generally yield 20 ms less start-up delay than LRU (non-transcoding system). Comparable to their performance in byte hit ratio, TVO yields start-up latency similar to LRU and FVO incurs larger start-up latency than TEC-11 and TEC-2 when the relative cache size is less than 40%.

2) Various Access Patterns: We now study the performance of caching algorithms in various network environments. In these simulations, similar parameters in the testing environment from above are used with the following differences. First, the access trace includes requests to three versions of 512, 256, and 128 kbps (labeled as version 0, 1, and 2, or b_0, b_1, and b_2). Note that there are requests to the full version (b_0) in this scenario. Second, we model the access to different versions as follows. Assuming the accesses to different versions of the objects follow a normal distribution with mean m, which is also the label of the dominant version, we study how the user access behavior and network environment affect the performance of caching algorithms. The probability of a

[2]FVO and TVO can also serve 0g streams. However, the full version is not requested from the clients in this simulation scenario. FVO caches the full version but always serves the transcoded version. TVO always caches the transcoded version, thus always serves 1g streams. We later show results from simulations where the full version is requested.

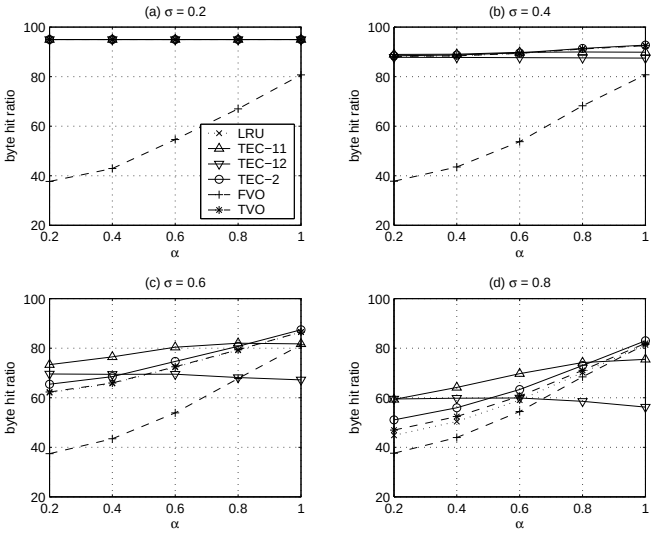

Fig. 6. Performance comparison of algorithms with fixed σ and cache size.

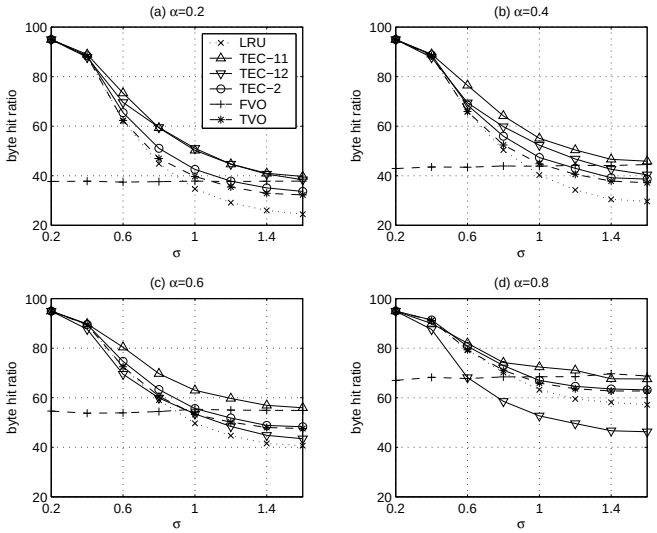

Fig. 7. Performance comparison of algorithms with fixed α and cache size.

version x getting accessed is:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-m)^2/2\sigma^2}, \quad (1)$$

where m and σ respectively represent the mean and the variation of how the versions are accessed. A small σ indicates that most of the accesses are to one version (m) and a large σ indicates that the accesses to different version are evenly distributed. Using the above access distribution model, we conduct the following experiments. We set $m = 2$ (i.e., b_2 is the dominant version when σ is small) and vary σ and α (of Zipf distribution). The relative cache size is fixed at 20%.

Figures 6 and 7 show that when σ is small (i.e., we have a homogeneous access pattern, with most accesses to b_2, the lowest bit-rate version of an object), the TEC algorithms and TVO use the cache space more efficiently, and hence have a high byte hit ratio irrespective of the change in popularity. However as σ increases, there is more heterogeneous accesses; b_0, b_1 and b_2 are more evenly accessed. Versions b_0 and b_1 are larger in size than b_2, which puts more pressure on the cache and hence the byte hit ratio of the TEC algorithms and TVO drop. FVO is immune to the change in σ since it always caches the b_0, the full version. As α increases, fewer objects become more popular. This helps FVO which can fit more objects in the cache (whose size is confined to 20% as in a practical setting) and attain higher byte hit ratio.

We also observe that when σ becomes larger, TEC-11 and TEC-2 achieve better byte hit ratio than other algorithms. TEC-2 outperforms TEC-11 when α increases, especially when σ value is large. Note that TEC-2 requires less transcoding load than TEC-11 because it produces more exact hits. This indicates that caching multiple versions achieves the best performance when accesses to the versions show high temporal locality.

Figure 7 shows that when α is small, TEC-12 performs similar to TEC-11 regardless of the value of σ. Moreover, TEC-12 requires less transcoding load since the accesses are mostly to the lowest bit-rate version. TEC-12 is hence the best strategy in this scenario. When α increases however, TEC-11 improves its performance more than TEC-12, especially when σ increases as well. This result shows that caching transcodable version achieves better byte hit ratio when object accesses are skewed on popularity, but evenly distributed across versions of the same object.

These analyses motivate a careful study of the user request patterns from the client group when selecting caching algorithms for the TeC proxy. We can select one of the TEC algorithms based on the access pattern. Alternatively, we can design an adaptive scheme that dynamically switches between different algorithms based on the analysis to maximize performance.

B. Enterprise-Trace-Driven Simulation

We have so far evaluated the algorithm performance using synthesized traces. We now use an actual media trace from a corporate environment.

1) Trace Analysis: The media server logs provided as input to our simulator are obtained from the servers of HP Corporate Media Solutions. We use log entries from April 1 through May 31, 2001. There were two servers running Windows Media Server (TM) that provide content to HP intranet clients around the world. The contents include audio and video coverage of keynote speeches at various corporate and industry events, messages from the company's management, product announcements, training video, and professional development courses. These servers have only a small fraction of their video content encoded at both high bit-rate and low bit-rate. Therefore, the bit-rate variation, though limited, helps evaluating the performance of TeC algorithms. Although the Windows Media Server is not RTP/RTSP-based, the client access pattern and the video popularity statistics extracted from the server logs are nevertheless useful.

To analyze the obtained traces, we categorize the video objects into one of seven categories. Categories C0, C1 and C2 contain objects with one version. C0 contains objects encoded

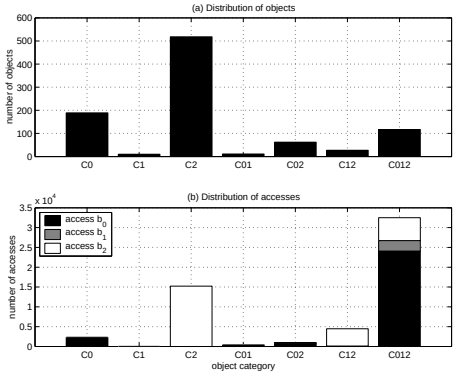

Fig. 8. Workload variance characteristics.

TABLE I

STATISTICS FOR MEDIA OBJECT BY VERSIONS.

Version	# of objects	# of accesses	Total size	Average size
$\sim$ 28 kbps	708	25,543	4,614 MB	7 MB
$\sim$ 56 kbps	165	2,839	3,049 MB	18 MB
$\sim$ 112 kbps	395	27,503	12,749 MB	32 MB
Total	1,268	55,885	20,412 MB	

at high bit-rate (b_0, above 100 kbps), C1 for mid bit-rate (b_1, 56 kbps), and C2 for low bit-rate (b_2, 28 kbps). Category C01 contains objects with two versions, at bit-rates b_0 and b_1. Similar definitions are derived for categories C02 and C12. Category C012 contains objects with all three versions. Figure 8 (a) shows the distribution of the number of objects in each category. Since the media server is targeting at corporate intranet users with similar network connection speed, the majority of the objects is coded in one version. There is a significant amount of audio clips encoded at 28 kbps, and hence a large number of objects categorized into C2. There are 115 objects coded in all three bit-rates (in category C012), and nearly 100 objects coded in various combinations of two versions (in categories C01, C02, and C12). The TeC system is the most effective when there are requests to these objects with multiple versions.

Figure 8 (b) shows the number of accesses to each category. Within each category, the number of accesses to different versions is also shown. Since the TeC system improves caching performance when there are accesses to multiple bit-rate versions, we focus on C01, C02, C12, and C012 categories. Note that the accesses to one version dominate in each of these categories. In most cases, the highest-bit-rate version is accessed the most often. This is an expected result as the accesses are mostly from the corporate intranet with high LAN bandwidth. Note also that the accesses to multiple versions are mainly to objects in C012. The accesses to version b_0 dominate in C012 with nearly 75% of the total access in this category. This indicates a pattern of homogeneous client access in corporate intranet environments.

Table I shows the statistics for each version: the number of unique objects and the total number of accesses to these objects. It also shows the total and average file size of these unique objects. We consider different versions of the same video content as independent objects and separately count the accesses to those objects.

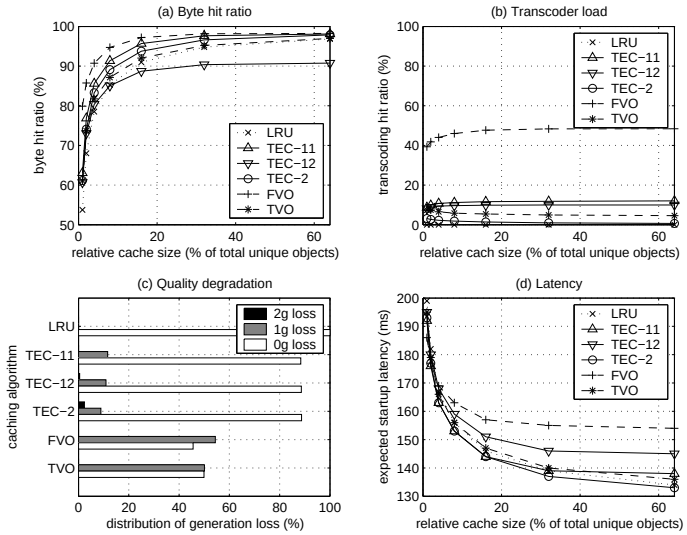

Fig. 9. Performance comparison in trace-driven simulations.

During the simulation, 20 GB of content resides on the origin server and there is a total of 27,926 accesses. The accesses show high temporal locality. Each day, there is on average 6 GB of live objects, which includes the new objects generated that day as well as objects from previous days that will get more accesses before the end of the measurement period.

2) Simulation Results: Figure 9 (a) shows the byte hit ratio. TEC-11 provides 3$\sim$10% byte hit ratio improvement over LRU. TEC-11 delivers this improvement only on accesses to lower bit-rate versions of objects where multiple versions are evenly accessed without one version predominantly being accessed. From Figure 8 (b), these are accesses b_1 and b_2 for category C012. They are only approximately 10% of the total number of accesses, and the majority of them are transcode hits. In an environment with more heterogeneous accesses, we expect significantly more accesses to be served from transcode hits, resulting in a better byte hit ratio improvement over LRU. Since the dominant version in an enterprise environment is most likely the highest bit-rate version, FVO shows the best performance, especially when the cache size is small. For the same reason, algorithm TEC-11 that caches higher bit-rate versions shows better result than TEC-12. TEC-2 achieves less improvement over LRU compared with TEC-11 because TEC-11 stores larger number of unique video objects. Nevertheless, as shown in Figure 9 (b), the transcoding load is significantly lighter in TEC-2 since it caches multiple versions of the same object. We also observe that the transcoding load of FVO is much heavier than other algorithms. Compared with FVO, TEC-11 achieves similar overall byte hit ratio with less transcoding load.

Figure 9 (c) shows the distribution of generation loss when the relative cache size is 16%. About 10% of the requests are served by TEC schemes with generation losses. For TEC-2, only 2% of the requests are served with two generation losses. As for FVO and TVO, the requests to the full version produce

0g streams, either from the cache or the origin. The requests to other versions are always served by transcoding from the full version, and hence 1g streams are produced.

Figure 9 (d) shows the expected latency, again using the values obtained from Figure 10 (b) of the Appendix. FVO caches full version only which results in more transcode hits, but on the other hand, it also results in fewer exact hits. Therefore, FVO shows the largest delay even though it achieves the highest byte hit ratio. Among TECs, TEC-12 shows the largest delay as most of the user requests are to higher bit-rate versions.

C. Summary

We evaluated the performance of TEC algorithms by simulating them in various network scenarios and comparing them with the traditional network caches without transcoding capability, and with FVO and TVO that also combine streaming media caching with transcoding.

Our simulation results indicate that compared with the traditional network caches, with marginal transcoding load, TeC improves the cache effectiveness, decreases the user-perceived latency, and reduces the traffic between the proxy and the content origin server.

When compared with FVO and TVO, the TEC algorithms generally showed better performance with less transcoding load. When the cache size is large or full versions are requested more often, FVO achieves the highest byte hit ratio as it stores more full version objects and serves requests to lower bit-rate versions by transcoding. However, it comes with the price of high transcoding load.

TEC and TVO put less burden on the transcoder than FVO. The performance of TVO could not match that of TECs however, as it does not support transcoding from an already transcoded object in the cache. Hence TVO requires more traffic between the content origin and the transcoding proxy than TECs. In addition, TVO may cause the cache to store multiple versions of the same video in an undeterministic fashion. On the other hand, TEC-11 and TEC-12 store at most one version per object.

Among TECs, TEC-2 gave good performances when the cache size is large, users are in heterogeneous environment, and the access pattern shows strong temporal locality. Compared with other TECs, TEC-11 generally showed the highest byte-hit ratio but also had the highest transcoding load.

IV. Related Work

Many researchers have contemplated with the idea of putting transcoders at network intermediaries. Transcoding of HTML and image contents is explored in [12], [13] but they do not cover streaming media transcoding. MOWSER [14] allows mobile users to specify the QoS parameters. Proxy agents between the web server and mobile users transcode the Web content into the viewing preference of the clients. A similar work [15] uses InfoPyramid data model to adapt web contents to mobile client capabilities. On-the-fly transformation of web contents at the network infrastructure is proposed in [16]. Lossy compression is used for each specific data-types to adapt to network and client variations. A video gateway [17] performs transcoding and rate adaptation for video transmission in heterogeneous environments. Although all of the above mentioned work investigates transcoding, none considers caching.

There are several recent studies that propose caching algorithms for transcoding proxies. Soft caching [18], NetBlitz [19], and TranSquid [20] are transcoding enabled proxy systems that propose caching, but they focus only on web images and not streaming videos. Minimal aggregate transcoding cost [21] is devised to cache multiple version in the transcoding proxy. This work takes additional factor such as transcoding delay into account in their cache replacement algorithm. This transcoding delay does not apply in our case since TEC proxy performs “streamlined” transcoding. A transcoding caching algorithm is devised for hierarchical cache networks in [22]. It focuses on the load sharing issues among hierarchy proxies. Again, [21] and [22] primarily consider transcoding of static web objects and not streaming video.

Several schemes for caching video streams from the Internet have been proposed. These schemes however, do not use transcoding at the local network proxies. Prefix caching [6] stores the initial parts of popular videos on the proxy to reduce the playback latency. It also considers smoothing techniques for VBR multimedia objects. Selective caching [23] stores only certain parts (not necessarily the initial part) of the video. The selection of the frames depends on network environments. Two algorithms are proposed, one each for QoS networks and the best-effort Internet. Partial caching with server scheduling is considered in [24] for saving bandwidth from the origin server. Partial caching in dynamic network conditions is studied in [25]. MiddleMan [26] aggregates a number of proxy caches connected within a network to cache video files. The system proposed in [27] has proxies perform request aggregation, prefix caching, and rate control to cache streaming media. Video staging [28] prefetches certain videos onto the proxies to preserve WAN bandwidth. Caching media streams using video summarization is proposed in [29]. In Mocha [30], a layered caching scheme is introduced to adjust stream quality based on per-layer popularity. The layering techniques are the most similar in philosophy to our approach, but as stated in Section I, layered-encoded formats may not always be available. An interesting work that analytically compares caching layered videos with caching multiple different video versions is found in [31].

V. Conclusion

We proposed the Transcoding-enabled Caching (TeC) system for streaming media distribution networks. By placing the transcoding service at the proxies, we perform content adaptation at the network intermediaries. The TeC proxies perform transcoding in addition to caching. TeC is useful for networks with heterogeneous environments as it transcodes the streaming objects to appropriate formats based on user’s networking and computing capabilities.

Given that most of the current media content on the Internet is coded in single-layered format, we believe adding

TABLE II
2G LOSS: PSNR DIFFERENCE (DB).

	raw-128		512-128	
	avg.	std.	avg.	std.
akiyo	0.93	0.32	0.5	0.2
hallw	0.98	0.21	0.14	0.12
coast	0.86	0.3	0.03	0.07
table	0.93	0.33	0.01	0.13

transcoding to the caching proxies brings caching effectiveness and efficiency. Our simulation results showed that combining caching with transcoding at the network intermediaries is a better choice than performing caching only, or providing multiple formats at the server. We proposed various caching strategies for our TeC system. The effectiveness of each algorithm strongly depends on user access pattern and network environment. In an corporate environment where users show less heterogeneity, caching one version for each video object is desirable. When the users have heterogeneous environment and the access pattern has strong temporal locality, caching multiple versions of popular objects is beneficial.

APPENDIX
TRANSCODER PERFORMANCE

We further quantitatively evaluate the 2g loss caused by repeated transcoding that may occur in the proposed TeC system. We use four MPEG-1 CIF resolution test sequences that are coded at 512 kbps with the GOP size of 12 and the structure of IBBPBB. Each sequence is transcoded to 256 kbps and then to 128 kbps. The transcoded video obtained thus suffers two generation losses. The original frames are used as references to compute the PSNR of the transcoded video.

Table II shows the average and standard deviation of the 2g loss in the PSNR differences for all four test sequences. Two PSNR differences are shown. One is the difference against the version that is directly coded from the raw frames (raw-128) and the other is against the version that is transcoded from the 512 kbps version (512-128). The 2g loss is larger when compared with the raw-128 version than with 512-128. However, to avoid the larger 2g loss, the content origin server must create and store 128 kbps version. The results verify that less than 1dB loss in PSNR is observed even for 2g loss.

We now evaluate the computing load and start-up delay of our TeC system. We setup a server-proxy-client testing system with intranet LAN connections, all in the same subnet. The origin server is implemented as an RTP/RTSP streaming server which delivers content at its coded rate. The platform is HP NetServer lp1000r with two 1.4 GHz Pentium III processors and 1 GB RAM running SuSE Linux 8.0. The TeC proxy also runs RTP/RTSP and it receives RTP packets from the origin server, transcodes on a frame-by-frame basis, and streams it to the client at the same time. The transcoding proxy runs on an HP workstation x4000, with two 2 GHz Intel Xeon processors, 512 kB L2-cache and 1 GB RAM running SuSE Linux 8.0. The same platform is used for the client that issues a group transcoding requests and dumps the received packets.

We use a sequence of 15 requests with five seconds apart, requesting for transcoding services on a test video sequence.

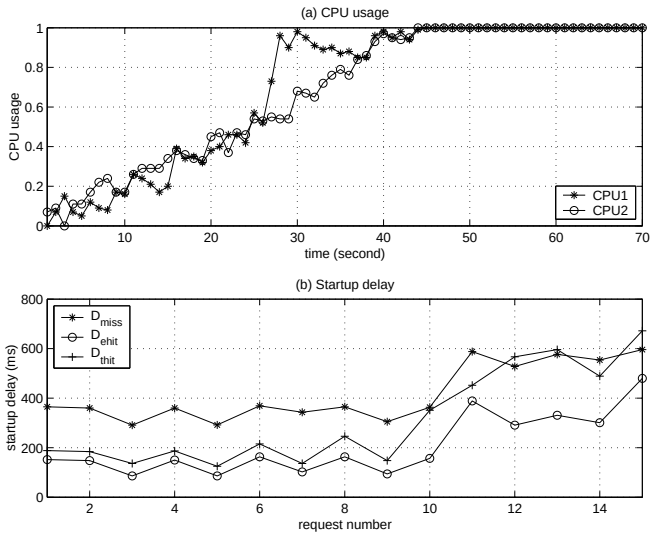

Fig. 10. CPU usage and delay.

The test stream is a CIF MPEG-4 system stream with audio and video coded at 16 kbps and 316 kbps respectively. The test sequence is coded at 15 Hz and is long enough (45 minutes) so that the number of concurrent transcoding sessions accumulates for each request arrival. On receiving each client request, the transcoder establishes an RTP session with the server and receives the original stream from it. The transcoding proxy then transcodes the video packets while relaying the audio packets without transcoding. The video bit rate is reduced to 100 kbps using the transcoder that we implemented according to Figure 2. The transcoding is performed one frame at a time. RTP packet timestamps in the origin stream are used for the clients to synchronize the received audio and video packets. The transcoder also keeps track of its own timestamp so that when the transcoding is lagging behind, frames are dropped in order to keep pace with the relaying of the audio packets (which are not transcoded).

During the test period, we use a UNIX command `top` to record the CPU and memory consumption with the update frequency of 1 Hz. The CPU load of the two CPUs in the transcoder platform are plotted in Figure 10 (a). Note that the system reaches the full usage after more than nine concurrent sessions. This is also the time the transcoding proxy starts to report frame drops. Similar tests carried out for QCIF resolution videos reveal that the number of concurrent sessions the proxy can handle is much larger, at the range of 20. The transcoder load per session can be further reduced if we consider using the multimedia extensions of the processor's instruction set (MMX) or lookup-table-based implementation. In addition, the ability of handling concurrent sessions can be further scaled up by using transcoder clusters.

Figure 10 (b) plots the start-up delay for each transcoding requests. Before the maximum transcoding capacity is reached, the average start-up delay for a miss (D_{miss}), an exact hit (D_{ehit}), and a transcode hit (D_{thit}) are 339 ms, 127 ms, and 173 ms, respectively. When QCIF resolution video (120 kbps) is used, all the delay parameters are shorter (185 ms, 35 ms, and 42 ms respectively), as the delay in delivery and transcod-

ing is much shorter when the video resolution is low.

REFERENCES

[1] J. Wang, "A survey of web caching schemes for the internet," *ACM Computer Communication Review*, vol. 29, no. 5, pp. 36–46, Oct. 1999.

[2] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Processing Mag.*, vol. 20, no. 2, pp. 18–29, Mar. 2003.

[3] B. Shen and S. Roy, "Architectural and algorithmic optimizations for down-scale transcoding of compressed video," HP Laboratories, Technical Report HPL-2001-258R1, Oct. 2001.

[4] E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," in *Proc. ACM SIGMOD'93*, Washington, DC, May 1993, pp. 297–306.

[5] S. Jin and A. Bestavros, "Greedydual web caching algorithm: Exploiting the two sources of temporal locality in web request streams," *Computer Communications*, vol. 24, no. 2, pp. 174–183, Feb. 2001.

[6] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE INFOCOM'99*, New York, NY, Mar. 1999, pp. 1310–1319.

[7] S. Chen, B. Shen, S. Wee, and X. Zhang, "Adaptive and lazy segmentation based proxy caching for streaming media delivery," in *Proc. ACM NOSSDAV 2003*, Monterey, CA, June 2003, pp. 22–32.

[8] X. Tang, F. Zhang, and S. Chanson, "Streaming media caching algorithms for transcoding proxies," in *Proc. ICPP 2002*, Vancouver, Canada, Aug. 2002.

[9] B. Shen, S.-J. Lee, and S. Basu, "Streaming media caching with transcoding-enabled proxies," HP Laboratories, Technical Report HPL-2002-210R1, Aug. 2002.

[10] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy, "Measurement and analysis of a streaming-media workload," in *Proc. USITS 2001*, San Francisco, CA, Mar. 2001.

[11] P. A. A. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 8, pp. 953–967, Dec. 1998.

[12] IBM websphere. [Online]. Available: http://www.developer.ibm.com/websphere/index.html

[13] S. Chandra, C. S. Ellis, and A. Vahdat, "Application-level differentiated multimedia web services using quality aware transcoding," *IEEE J. Select. Areas Commun.*, vol. 18, no. 12, pp. 2544–2565, Dec. 2000.

[14] H. Bharadvaj, A. Joshi, and S. Auephanwiriyakul, "An active transcoding proxy to support mobile web access," in *Proc. IEEE SRDS'98*, West Lafayette, IN, Oct. 1998, pp. 118–123.

[15] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting multimedia internet content for universal access," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 104–114, Mar. 1999.

[16] A. Fox, S. D. Cribble, Y. Chawathe, and E. A. Brewer, "Adapting to network and client variation using infrastructural proxies: Lessons and perspectives," *IEEE Personal Commun. Mag.*, vol. 5, no. 5, pp. 10–19, Aug. 1998.

[17] E. Amir, S. McCanne, and H. Zhang, "An application level video gateway," in *Proc. ACM Multimedia'95*, San Francisco, CA, Nov. 1995, pp. 255–265.

[18] J. Kangasharju, Y. G. Kwon, and A. Ortega, "Design and implementation of a soft caching proxy," *Computer Networks and ISDN Systems*, vol. 30, no. 22-23, pp. 2113–2122, Nov. 1998.

[19] S. Acharya, H. Korth, and V. Poosala, "Systematic multiresolution and its application to the world wide web," in *Proc. IEEE ICDE'99*, Sydney, Australia, Mar. 1999, pp. 40–49.

[20] A. Maheshwari, A. Sharma, K. Ramamrithan, and P. Shenoy, "TranSquid: Transcoding and caching proxy for heterogeneous e-commerce environments," in *Proc. IEEE RIDE 2002*, San Jose, CA, Feb. 2002.

[21] C.-Y. Chang and M.-S. Chen, "Exploring aggregate effect with weighted transcoding graphs for effcient cache replacement in transcoding proxies," in *Proc. IEEE ICDE 2002*, San Jose, CA, Feb. 2002, pp. 383–392.

[22] V. Cardellini, P. S. Yu, and Y.-W. Huang, "Collaborative proxy system for distributed web content transcoding," in *Proc. ACM CIKM 2000*, McLean, VA, Nov. 2000, pp. 520–527.

[23] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE J. Select. Areas Commun.*, vol. 20, no. 7, pp. 1315–1327, Sept. 2002.

[24] O. Verscheure, C. Venkatramani, P. Frossard, and L. Amini, "Joint server scheduling and proxy caching for video delivery," *Computer Communications*, vol. 25, no. 4, pp. 413–423, Mar. 2002.

[25] S. Jin, A. Bestavros, and A. Iyengar, "Accelerating internet streaming media delivery using network-aware partial caching," in *Proc. IEEE ICDCS 2002*, Vienna, Austria, July 2002, pp. 153–160.

[26] S. Acharya and B. Smith, "MiddleMan: A video caching proxy server," in *Proc. NOSSDAV 2000*, Chapel Hill, NC, June 2000.

[27] E. Bommaiah, K. Guo, M. Hofmann, and S. Paul, "Design and implementation of a caching system for streaming media over the internet," in *Proc. IEEE RTAS 2000*, Washington, DC, June 2000, pp. 111–121.

[28] Z.-L. 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.

[29] S.-J. Lee, W.-Y. Ma, and B. Shen, "An interactive video delivery and caching system using video summarization," *Computer Communications*, vol. 25, no. 4, pp. 424–435, Mar. 2002.

[30] R. Rejaie and J. Kangasharju, "Mocha: A quality adaptive multimedia proxy cache for internet streaming," in *Proc. ACM NOSSDAV 2001*, Port Jefferson, NY, June 2001.

[31] F. Hartanto, J. Kangasharju, M. Reisslein, and K. W. Ross, "Caching video objects: Layers vs versions?" in *Proc. IEEE ICME 2002*, Lausanne, Switzerland, Aug. 2002.