Streaming Media Caching with Transcoding-Enabled Proxies

Bo Shen, Sung-Ju Lee, Sujoy Basu
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto

HPL-2002-210 (R.1)
October 31st , 2003*

E-mail: {boshen, sjlee, basus}@hpl.hp.com

proxy caching, streaming media caching, video transcoding

We propose a transcoding-enabled streaming media caching system (TeC) along with a new set of caching strategies. Our system is designed for efficient delivery of rich media web contents to heterogeneous network environments and client capabilities. The proxies perform transcoding as well as caching in our system. This design choice allows content adaptation to be performed at the edges of the networks. 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 serving the transcoded video directly from the proxy, we improve the cache performance. Performance evaluation via simulation is presented. Specifically, simulations using both synthesized traces and real traces derived from enterprise media server logs are conducted. Simulation results indicate that by incorporating transcoding capability at the network edges, the traffic to the content origin server is further reduced.

* Internal Accession Date Only Approved for External Publication
Published in the proceedings of the 6th IASTED International Conference on Internet and Multimedia Systems and Applications, 12-14 August 2002, Kauai, Hawaii.

Streaming Media Caching with Transcoding-Enabled Proxies

Bo Shen, Sung-Ju Lee, and Sujoy Basu
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA
{boshen, sjlee, basus}@hpl.hp.com

Abstract

We propose a transcoding-enabled streaming media caching system (TeC) along with a new set of caching strategies. Our system is designed for efficient delivery of rich media web contents to heterogeneous network environments and client capabilities. The proxies perform transcoding as well as caching in our system. This design choice allows content adaptation to be performed at the edges of the networks. 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 serving the transcoded video directly from the proxy, we improve the cache performance. Performance evaluation via simulation is presented. Specifically, simulations using both synthesized traces and real traces derived from enterprise media server logs are conducted. Simulation results indicate that by incorporating transcoding capability at the network edges, the traffic to the content origin server is further reduced.

Keywords: proxy caching, streaming media caching, video transcoding.

1 Introduction

Delivery of streaming media contents over the Internet is a challenge; encoding, delivery, caching, and processing are far more difficult than those of simple web objects. For example, when CNN.com posts a news video clip on its Web server, it first encodes the news clip at several different bit-rates (28-56 Kbps for dial-up connections and 150-plus Kbps for broadband networks [8]) to satisfy users with different network connection speeds. In addition, the prosperity of wireless network brings more heterogeneous client devices (e.g., laptops, PDAs, palm pilots, cell phones, etc.). Traditional caching system treats each client request equally and independently. Usually, popular items are cached at a proxy close to the end user and therefore, it reduces the traffic between the content origin and proxies as well as the user perceived startup latency. However, various different bit-rate versions of the same video clip may be cached at the proxy at one instance, which can be a waste of storage.

Our solution is to enable the caching proxy with the transcoding ability so that variants of a video object can be delivered by transcoding the cached content to the appropriate format, instead of accessing the object all the way from the content origin. This approach puts transcoding units in the content delivery path. Putting computing resources in the content delivery path has been addressed in [2] and [14]. The network intermediaries perform dynamic content adaptation in this approach. One of the advantages of having transcoding-enabled proxy is that the content origin servers need not generate different bit-rate videos. Moreover, heterogeneous clients with various network conditions will receive videos that are suited for their capabilities, as content adaptation is easily done at the network edges.

We investigate the possibility of using computing resource to trade off caching storage; therefore further improving the responsiveness for media-rich Web access. One may argue that storage is cheap these days and saving storage is not necessary. This is partially true. Moreover, The processing capabilities have been advanced and the processors are cheap as well. In addition, because the video files are very large in size, we cannot assume the unlimited availability of storage. Focusing on streaming video delivery over the Internet, we use video transcoders at the caching proxies. Video transcoding itself is a computing intensive task. Many works target at improving the efficiency of the task. In [18], we transcode DVD video at high resolution to lower resolution in real time. The bit-rate is also reduced during the transcoding.

For the delivery of multimedia content, many coding techniques have been developed to deal with heterogeneous network conditions and diverse client capabilities. Among these techniques, scalable coding, layered coding are the typical examples. These techniques are also included in some of the video coding standards such as MPEG-2 and MPEG-4. However, given the current distribution of multimedia objects on the Internet, majority if not all of the content is actually coded in non-scalable and single-layered format. We argue that using transcoding to adapt to the heterogeneous is a much more practical approach.

Given that the caching proxy is transcoding-enabled, new adaptive caching systems need to be developed for better utilization of the storage resource. This paper proposes a set of caching algorithms taking into account that variants of the same video object may exist in the system at any point of time. The variants are produced either a priori or on demand by the transcoder. The algorithm can choose to cache single or multiple variants of a video. The cached video version can be either from the origin server or generated by the transcoder. Transcoding can be used to trade off the origin server access. We evaluate our proposed TeC (Transcoding-enabled Caching) system using various experiments, including trace-driven simulation. A trace analysis on variants of a streaming video object in a corporate network environment is provided. Our results indicate that our system shows better caching performance than the traditional caching system with manageable computation load on the transcoder.

The rest of the paper is organized as follows. In Section 2, we propose the system architecture of the transcoding-enabled caching proxy along with a set of caching algorithms specifically designed for TeC. Performance analysis and trace-driven simulation results are presented in Section 3. Specifically, simulations based on proxy traces derived from a real cooperate media server log and simulations based on synthesized traces are conducted. Related work is surveyed in Section 4 and we conclude in Section 5.

2 Tracoding-enabled Caching

2.1 System Architecture

Proxy caches are often deployed at the edges of the network to reduce the traffic to and from the origin server and user perceived latency. We propose a transcoding enabled caching proxy that serves various bit-rate versions of video objects to the end users with different devices or connection profiles. Focusing on the video delivery, we illustrate the system architecture as following.

Transcoding-enabled caching (TeC) proxy consists of the components as shown in Figure 1. The proxy acts as a client to the content server. A RTP/RTSP client is therefore built into the proxy to receive the streamed content from the origin server (uplink). The received stream is put into the incoming buffer. The transcoder continuously pulls bit streams from the incoming buffer and subsequently pushes the transcoded bits out to the outgoing buffer. The proxy decides to cache the content either from the incoming buffer or the outgoing buffer while it is being produced by the transcoder. Additionally, The proxy acts as a server to the end user. Therefore, a RTP/RTSP server is built to stream the video to the end user (downlink). The data in the outgoing buffer is obtained either from the transcoder or from the caching system.

The size of the incoming buffer and the outgoing buffer 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. As long as the transcoding bit-rate is larger than the minimum of the uplink and the downlink bandwidths, the transcoding process does not significantly increase the end-to-end delay. Nevertheless, video transcoding can be computing intensive. Many works are under investigation to reduce the workload of such a session. Among those, compressed domain based approach provides the best performance [18]. In compressed domain transcoding, the incoming video is only partially decompressed. Rate adapting is performed in the compressed domain while the motion information is reused. This approach considerably improves the speed over the conventional decoding-transcoding-recoding approach. While not in the scope of this paper, we assume the transcoder is capable of handling reasonable number of concurrent sessions in real time. This allows us to focus on the investigation of the caching benefits the collocated transcoder brings.

Given the real time transcoding capability, the TeC proxies can dynamically transcode video objects to different variants to satisfy the end users in heterogeneous networks. Each variant is a version. If version x can be obtained by transcoding from 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 degradation in fidelity comparing 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 resolution already reduces the bit-rate by a factor of approximately four.

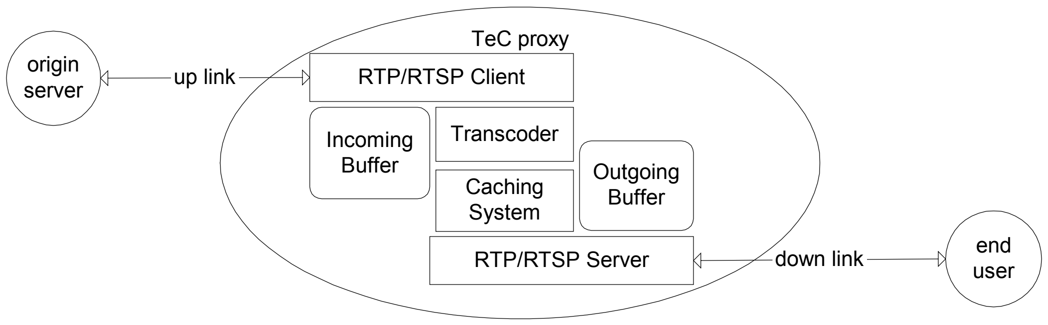


Figure 1. System and components for transcoding-enabled caching proxy.

Note that the variation in versions delivered to the end user is not transparent to the end user. Either the end user specifically asks for certain version of an object based on his/her awareness of his/her connection and display device, or agent software of the streaming client tells the proxy its connection and device capability parameters and the proxy then choose one version for the session. In addition, the TeC proxy is often at the edge of the network close to the end user, it is not meant for adapting to the bandwidth variation in up link. If there are packet losses in up link due to congestion, the TeC proxy can choose not to cache for the session.

2.2 Caching Algorithms

A TeC proxy trades off computation with storage. The main idea is to serve the end user with the appropriately transcoded version of the cached video whenever possible, depending on the network capacity and connection profile of the user. Let us assume that the origin server has n versions at bit-rates b_1, b_2, …, b_n for each video object. The highest bit-rate version is b_1 and the lowest is b_n, i.e., $b_1 > b_2 >\ldots> b_n$. When version b_i is requested from the end user, and if there is version b_j ($b_j > b_i$, i.e., b_j is a transcodable version for b_i) in cache, the TeC proxy transcodes b_j to b_i instead of fetching b_i from the content origin. Therefore, it is a cache hit even though b_i is not directly available from the cache. 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 does.
- *Miss*, the requested or a transcodable version of the video does not exist in the cache.

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 stream it to the user. Hence, our system works well regardless of whether the content origin supports various bit-rate versions of the videos or not. We propose two types of caching algorithms that behave differently when each event occurs.

2.2.1 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 utilize the storage space efficiently. 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 for the object and streams it to the end user. If a request leads to a cache miss, the TeC proxy fetches the video from the origin server, streams it to the end 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 has to be responded from the content origin (i.e., the end user requests a higher bit-rate version than the cached one), then it is considered a cache miss.

In general, if the end user requests version b_i of a video while b_j, where $b_i > b_j$, exists in the cache, then 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 exist in the cache, the lower bit-rate version is evicted from the cache. If a video 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 can choose which version of the video 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 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_i of a video while b_j, where $b_i < b_j$, exists in the cache, b_i is transcoded from b_j and streamed to the user. TEC_11 refreshes the access record of b_j , but TEC_12 removes b_j and caches b_i .

Whenever the cache becomes full and requests to the un-cached video are received, certain files in the cache must be replaced. We simply use the existing popular cache replacement algorithms (e.g., LRU, LFU, LRU-*k* [15], or GD* [12]) for this purpose. The pseudo code of algorithms TEC_11 and TEC_12 is presented in Figure 2.

```
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 version $b_j$ of the same video 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 TEC_11
        update the access record of $b_j$
    if TEC_12
        remove $b_j$ from the cache
        store $b_i$ in the proxy cache
        update the access record of $b_i$
else if $b_j$ is in the proxy cache and $b_j < b_i$
    remove $b_j$ from the cache
    fetch version $b_i$ of the video from the content origin
    stream $b_i$ to the end user
    if the size of $b_i$ is larger than the available cache space
        select the replacement victim using the selected algorithm
    store $b_i$ in the proxy cache
    update the access record of $b_i$
else
    fetch version $b_i$ of the video from the content origin
    stream $b_i$ to the end user
    if the size of $b_i$ is larger than the available cache space
        select the replacement victim using the selected algorithm
    store $b_i$ in the proxy cache
    update the access record of $b_i$
```

Figure 2. Cache single version, algorithm TEC_11 and TEC_12.

2.2.2 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_1 and b_2 are both in the cache, a request to b_2 will lead to an exact hit, i.e., no transcoding is needed. In addition, if the temporal-locality of accesses to a certain video object across its variants is high, this approach may further improve the caching efficiency.

In this algorithm, 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 end user and caches it even if 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 request version b_i of a video while b_j ($b_j > b_i$) exists in the cache, b_i is transcoded from b_j, delivered to the user and cached. Note that b_i and b_j now both exist in the cache.

Similar to TEC_11 and TEC_12, when the space is needed to cache new objects, we use an existing cache replacement algorithm. For example, if LRU is used, TeC proxy will find the least recently used objects and replace them with either the fetched or the transcoded version. Figure 3 shows the pseudo code of TEC_2 algorithm.

```
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 user from the cache
        if the size of $b_i$ is larger than the available space
                select the replacement victim using the selected algorithm
        store $b_i$ in the proxy cache
        update the access record of $b_i$ and $b_j$
else
        fetch version $b_i$ of the video from the content origin
        stream $b_i$ to the end user
        if the size of $b_i$ is larger than the available cache space
                select the replacement victim using selected algorithm
        store $b_i$ in the proxy cache
        update the access record of $b_i$
```

Figure 3. Cache multiple versions, algorithm TEC_2.

2.2.3 Discussion

The effectiveness of the three presented algorithms highly depends on the user access behavior and the network environment of those users. For instance, when the users connected to a proxy have similar network capacities (e.g., employees in a corporate that have high-speed network connection during work hours), 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 (e.g., some are using PDAs while on the road, some are dialing-up to the Internet from home, and some are using DSL or cable modem to connect to the network) and the access behavior shows strong temporal locality, TEC_2 will show superior performance. In the next section, we simulate our proposed algorithms for performance evaluation.

3 Simulations and Results

We developed a stack-based implementation is developed for the performance evaluation simulation study. To focus on evaluating the benefit of TeC scheme, no prefix caching [16] is considered. That is, if an object is cached, the whole object is cached. In practice, TeC can work with prefix caching to provide better performance. We conduct two types of simulation; first, an enterprise trace is used to simulate prolonged access of media content in mostly homogenous network environment. To evaluate the performance in a more heterogeneous network, we then use a synthesized trace. In all the simulations, the TeC proxy uses LRU as the cache replacement algorithm. Based upon various cache capacities, we compare the performance of the TeC proxy with that of a regular caching proxy. The regular caching proxy only serves as a regular interception proxy using the same caching algorithm (LRU). That is, the proxy treats each access independently even if the access is to the same media object but a different version. We call this algorithm a "reference model".

3.1 Enterprise-Trace-Driven Simulation

The media server logs provided as input to our simulator are obtained from the servers of HP Corporate Media Solutions. All log entries are from April 1 through May 31, 2001. During those two months, there were two servers running Windows Media Server (TM) that serve content to clients around the world within HP intranet. 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 for employees. A detailed analysis of the overall characteristics of these logs (covering an earlier period of time) can be found in [6].

Note that the Windows Media Server is not RTP/RTSP-based as we would prefer. Nevertheless, the client access pattern and the video popularity statistics extracted from the server log are useful for our trace-driven simulation. For the TeC simulation, the server log cannot be used in its primitive form. In the following, we first describe how to extract proxy traces from the server logs and then provide an analysis that focuses on the variant issue on a media server.

To derive proxy traces from primitive server logs, we partition the server log into separate proxy access logs of four major geographical areas connecting the HP intranet: North America, Europe, Asia-Pacific Region and South America. For simplicity of this derivation, countries in Africa and Near-East, such as Egypt, Israel and Arab countries are included in Europe. We primarily use the domain name portion of the client's hostname, as recorded in the log entry, to assign the region. However, when the hostname field is empty, we try to resolve the hostname from the client's IP address. In rare cases when even that method fails, we use the language country-code of the client. This is not completely accurate, since users traveling to another country with their laptop may be accessing the HP intranet. We believe however that such accesses are mainly by salespersons covering some sales region, so they are unlikely to intersect the regions we defined.

Another preprocessing is needed to identify multiple bit-rate-variants of the same media object. When these variants are present in the media server, they are identifiable by some suffix of their URL. We find two common patterns. One is the use of the extensions *28.asf, *56.asf, and *110.asf or *112.asf. It is verified that these objects are coded at 28Kbps, 56Kbps , and above 100Kbps respectively. We label them as low (b2), medium (b1) and high (b0) bit rate variants, respectively. Another pattern is the use of *56k.asf and *100k.asf suffices. We label them as medium (b1) and high (b0) bit rate variants, respectively. Unfortunately, only a small number of the media objects have such variants already identifiable. Therefore, we further look at the average bandwidth field of each entry in the server logs. We label the access as to a low or medium bit rate variant if the bandwidth experienced while delivering the object is less than 28 Kbps or 56 Kbps, respectively. Otherwise, we label the access as to a high bit rate variant.

The final step of the preprocessing is to eliminate entries that do not fit the requirements of the simulator. For example, entries with the URL of the content absent, or for which the proxy assignment by region does not work are eliminated. Entries with timestamp not available or zero bytes transferred are also discarded.

To analyze the proxy traces we obtained from preprocessing, we define seven variant categories. Category c0, c1 and c2 contain objects with only one variant. That is, c0 contains objects encoded at high bit rate only. Category c1 contains objects encoded at medium bit rate only, and c2 for low bit rate only. Category c01 contains objects with two variants, at bit rate b_0 and b_1,. Similar definitions can be derived for categories c02 and c12. Category c012 contains objects with three variants. That is, for each object in category c012, there are three variants on the server. Using the extracted proxy trace for North America as an example, Figure 4 shows the distribution of the number of objects in each variant category. Since the media server is targeting at corporate intranet users, it is not surprising that the majority of the object are coded in one variant only. In addition, we see the spike in category c2 because there is a significant amount of audio clips encoded at 28Kbps. Since there is no variation in the accesses to objects in categories c0, c1 and c2, the TeC system will not improve the caching performance of these accesses. However, there are also 115 objects coded in all three bit rates (in category c012), and around 100 objects coded in various combinations of two bit rates (in categories c01, c02 and c12). The TeC system can improve the system performance for the accesses to these objects.

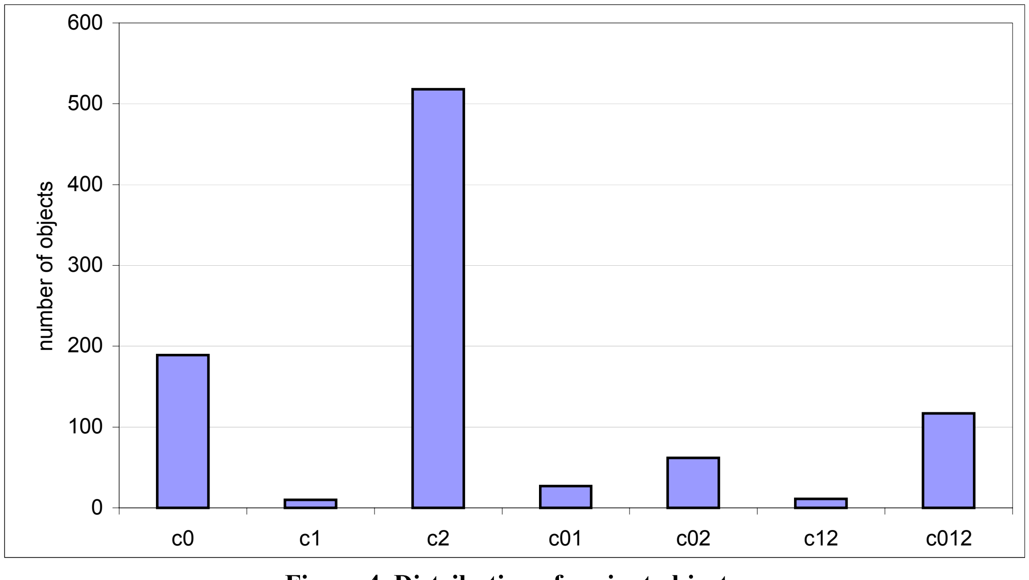


Figure 4. Distribution of variant objects.

Figure 5 shows the number of accesses within the investigated time period for each of the variant categories. Within each category, the number of accesses to different variants is also shown. Since the TeC system improves caching performance only when there are accesses to bit-rate variants, we pay our attention to c01, c02, c12 and c012 categories. Note that the access to one variant dominants in each of these categories. In most cases, the highest-bit-rate variant is accessed more often. This result is expected, as the accesses are mostly from the corporate intranet with high bandwidth. Also note that the accesses to variants are mainly to objects in category c012. Within this category, the access to variant b0 dominants. The access to other variants is nearly 25% of the total access to objects in the category, and less than 10% of total access overall. This indicates a pattern of access from clients in homogenous conditions, which is also expected in corporate intranet environments.

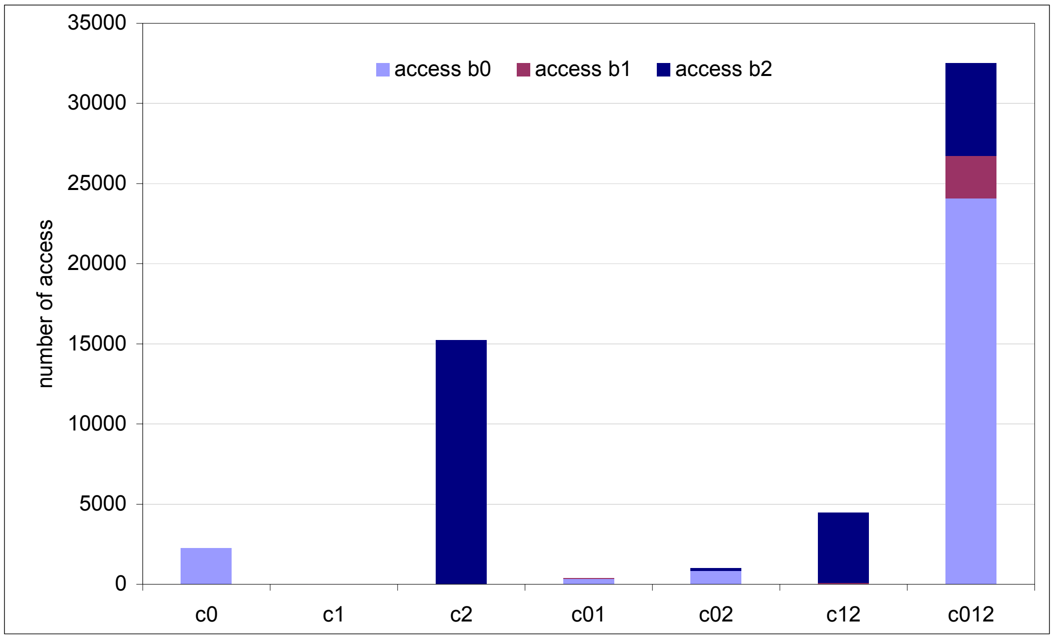


Figure 5. Distribution of accesses in variant categories.

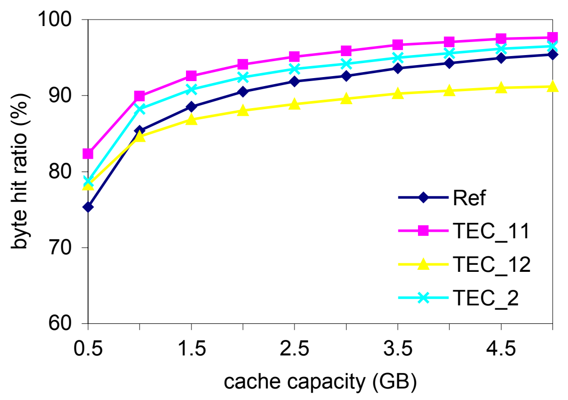
To give an overview of the content variation, Table 1 shows the statistics for each variant type: the number of unique objects and the total number of accesses to these objects. It also includes the total and average file size (in megabytes) of these unique objects. We consider each variant of the same content as independent objects and count the accesses to those objects separately.

Variant	Number of objects	Number of accesses	Total size (MB)	Average size (MB)
b0	395	27503	12749	32
b1	165	2839	3049	18
b2	708	25543	4614	7
Total	1268	55885	20412	

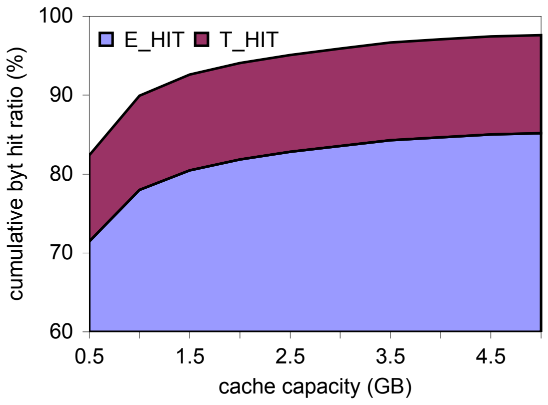
Table 1 Statistics for media objects by variant types

During the measured two months, there is at least 20 GB of content resides on the content server and there are total of 55885 accesses. However, the 20GB of data is not on the server from the beginning to the end. To obtain an understanding regarding the amount of content on the origin server in a smaller time frame, we perform the following analysis. For each content object, we find the first and the last day it is accessed within the measure period. In between, the object must be on the origin server. Collectively, we find that on average, there is at least 6GB of content on the server on any given day.

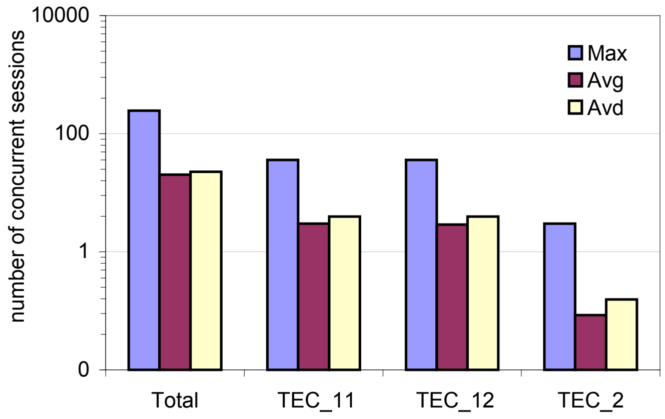
Figure 6 shows the simulation result using the proxy trace generated for North America region. Simulations using proxy traces generated for other regions provide similar results. Figure 6 (a) shows the overall byte hit ratio, (b) shows the contribution of transcode hit and exact hit for TEC_11, and (c) shows the statistics of the number of concurrent session (for TEC algorithms, it is the number of concurrent transcoding sessions) at each access timestamp when the cache capacity is 2GB (10% of total object size). Specifically, the maximum, average, and the average of the absolute deviation are shown.



(a)



(b)



(c)

Figure 6. Simulation results from media-server trace.

The simulation results show the caching performance improvement the TeC system brings when the access to media object is to almost homogenous variants. In this particular case, the proxy using TEC_11 provides 3~5% byte hit ratio improvements over the reference model. Considering that the access to variants is only about 10% of total access, the improvement indicates 30~50% of the variant-based access can be served from TeC directly therefore saving server bandwidth. Since the dominant version in enterprise environment is most likely the high bit-rate version, algorithm TEC_12 that caches lower bit-rate variants produces worse result. Note that TEC_2 achieves less improvement in byte hit ratio as compared with TEC_11. Nevertheless, since TEC_2 caches multiple versions of the same object, the transcoding load is reduced significantly. This is evident from Figure 6 (c). For the cache size at 10% of the total object size, the average concurrent transcoding session required for TEC_2 is orders of magnitude less than that for TEC_11.

3.2 Synthesized-Trace-Driven Simulation

The simulation in Section 3.1 showed the performance of a TeC proxy in a corporate intra network where less heterogeneity is expected. To evaluate the performance of the proposed system for a variant-based media website serving heterogeneous clients, we use the following "synthesized" trace as no public trace of this kind is available.

To generate the synthesized trace, we create a pool of 500 original video objects with their playing time varying from 5 to 15 minutes. The access pattern is characterized by the following three factors:

1. Popularity of the video: We assume a Zipf distribution with α of 0.47 [7].
2. Access arrival interval: We assume a random arrival through a Poisson process.
3. Variety of downlink capacity: Three settings are selected. We designate 512 Kbps and 256 Kbps for desktop users, and 128 Kbps for mobile users.

The variation of the downlink capacity also dictates the variants of the media object. Therefore, the size of each variant of a video object is the production of its length in seconds and its bit-rate. Using these parameters, we generate a content pool of approximately 30 GB of data.

Two different models of simulation are designed. First, we perform the simulation through an all-version-origin model. In this model, the original content server provides multiple bit-rate versions of each original video object. The client requested video is either served from the origin server, cache at the TeC proxy, or it is generated by the TeC proxy through transcoding. Second, we perform the simulation through a master-version-origin model where only one master version of each video object is available at the content origin server. In this model, the client requests either the master version or a lower bit-rate version of the video. The lower bit-rate variant is generated exclusively by the TeC proxy.

3.2.1 All Version Origin Model

We use an "even-variant" scenario that is defined as follows. In this scenario, media contents are accessed evenly by clients in each downlink bandwidth capacities. This scenario represents a highly heterogeneous environment. While the popularity of video objects follows a Zipf-like distribution (α=0.47), the downlink capacity varies evenly. The simulation lasts 8 hours with 600 accesses arriving at a random Poisson process. Consequently, the average access arrival time interval is 48 seconds. During the simulation, a total of 12GB of content is delivered to the clients.

Figure 7 (a) shows the byte hit ratio for various cache capacities in even-variant scenarios. Note that TEC_11 algorithm provides the best result, showing nearly 15% performance improvements over the reference model. Figure 7 (b) shows the portion of exact hit and transcode hit for the TEC_11 algorithm. The result indicates that from 20% to 50% of the bytes can be served through transcoding. Figure 7 (c) shows the total number of streaming sessions and the number of transcoding sessions for each caching algorithm when the cache capacity is 4 GB. The result from 1000-th second to 6000-th second of the 8-hour simulation is shown. The number of concurrent transcoding sessions is well within the range that a modern PC can handle.

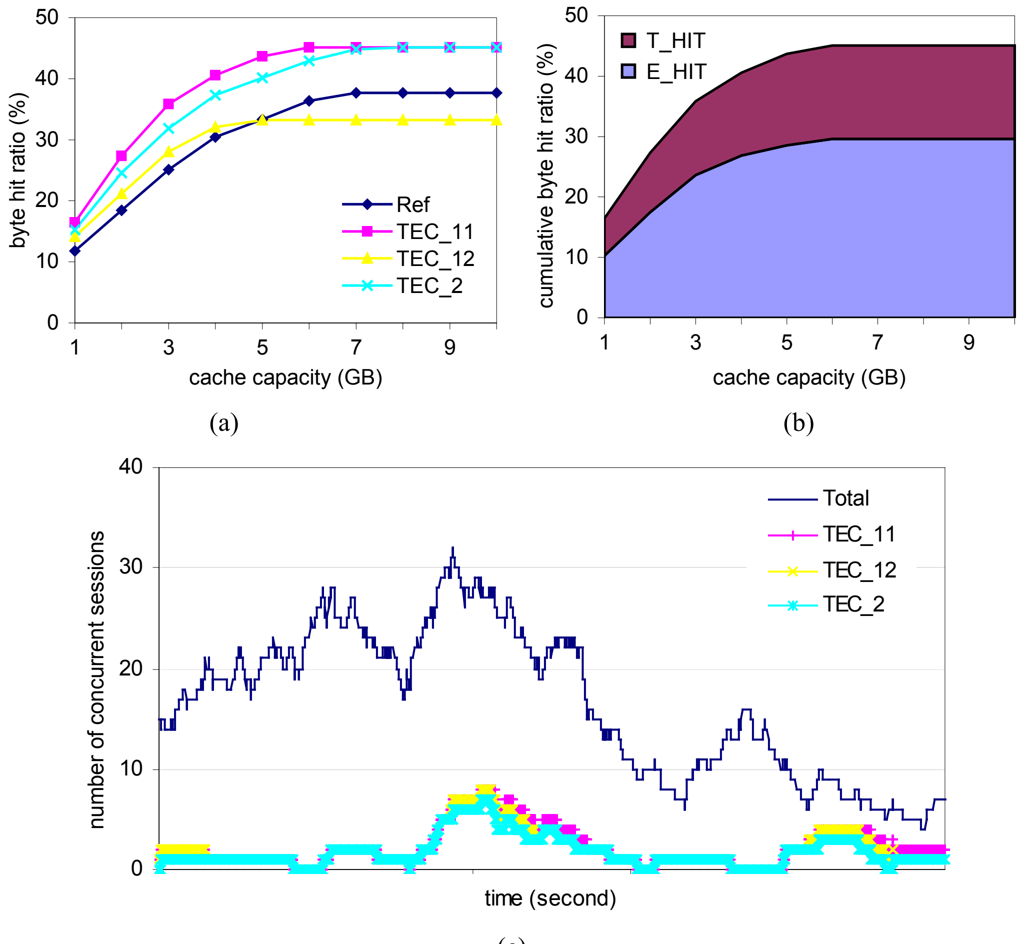


Figure 7. Simulation results for even-variant scenario.

3.2.2 Master Version Origin Model

In Section 3.2.1, we assumed that the origin content server provides multiple bit-rate versions of any video object. This may not be the case for many websites, such as a media website created specifically for high-bandwidth users. The TeC proxy can help increase the reach and the performance of these website by providing an on-the-fly transcoding and caching for localized access. To study the performance under this circumstance, we define a "master-version-origin" mode as follows. In this mode, only one master version is stored at the origin server. Therefore, when there is a cache miss, the master version is fetched always from the origin. However, the TeC proxy can choose to cache either the master version or the transcoded version. We again compare the performance of our algorithms with a reference model where a transcoder is not available and only the master version is cached at the proxy. We use a dominant-variant scenario in this experiment.

In dominant-variant scenario, we picture that the requests are mainly from mobile users after work checking news through cell phones or PDAs. The access arrival interval is short and highly temporally localized. We simulate this scenario by generating 600 accesses arriving in a random Poisson process throughout a one-hour simulation. Therefore, the average access arrival interval is six seconds. The variation in downlink capacity is relatively small; 90% of the downlink traffic is assigned to the dominant downlink capacity. We use 128 Kbps downlink as the dominant link. There are occasional requests from 256 Kbps or 512 Kbps downlinks. During the simulation, a total of 6 GB of data is delivered to the clients.

Figure 8 (a) shows the byte hit ratio. Our algorithms achieve 3~6% improvement over the traditional LRU system. When cache capacity is small, TEC_12 achieves better effectiveness than TEC_11. However, as we can see from Figure 8 (b), the performance gain primarily results from exact hits rather than transcode hits, since transcode hits account for 3~10% of the total byte hit.

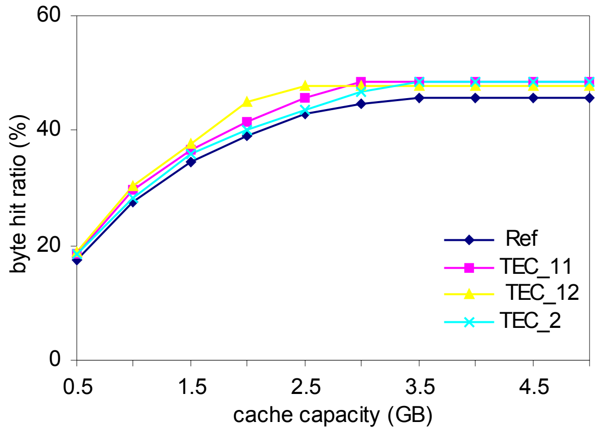
For requirement on computing resources, Figure 8 (c) shows the concurrent session load on the TeC proxy. The load on the transcoder is much heavier compared with Figure 7 (c) which shows the load of all-variant-origin model. On occasion, more than 50% of the total concurrent session needs transcoding, which poses a heavy load on the transcoding unit. This is due to the high density of client accesses and the predominant requests for low bit-rate variant. TEC_11 has a higher level of transcoding load than those of TEC_12 and TEC_2. This result is not surprising since the TEC_11 algorithm caches the highest bit-rate version, while TEC_12 and TEC_2 cache lower or multiple versions. Therefore, an exact hit is more likely in TEC_12 and TEC_2 than in TEC_11. Variations of the TEC algorithms provide different level of load on the transcoder. In practice, one can choose to use different TEC algorithms based on the available transcoding resource. For example, if a TeC proxy can handle limited concurrent transcoding sessions, TEC_2 is the best choice since it incurs less concurrent transcoding sessions as can be seen from Figure 8 (c).

4 Related Work

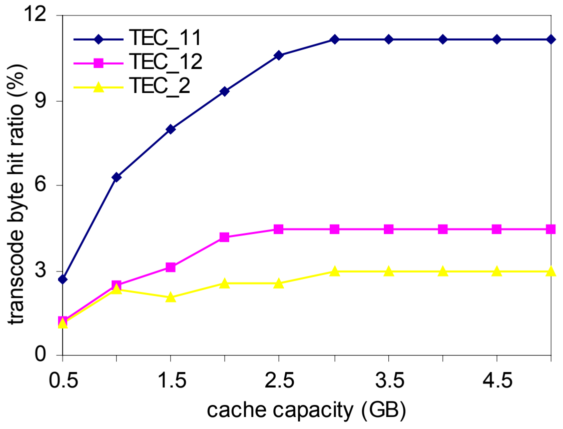
Several schemes for caching video streams from the Internet have been proposed. Prefix caching [17] stores the initial parts of popular videos on the proxy to reduce the playback latency. MiddleMan [1] aggregates a number of proxy caches connected within a network to cache video files. The system proposed in [4] has proxies perform request aggregation, prefix caching, and rate control to cache streaming media. Video staging [20] prefetches certain videos onto the proxies to preserve WAN bandwidth. None of the above schemes uses transcoding at the local network proxies.

Many works have contemplated with the idea of putting transcoders at network intermediaries [3,9,10,11,19]. MOWSER [3] 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. MOWSER however, does not deal with proxy caching. A similar work is done at [19] that uses InfoPyramid data model to adapt web contents to mobile client capabilities. Web caching is not considered in this work, either. On-the-fly transformation of web contents at the network infrastructure was proposed in [9]. Lossy compression is used for each specific data-types to adapt to network and client variations. An analytical framework for transcoding web streams at the proxies was discussed in [10]. IBM Websphere transcoding publisher [11] can transcode HTML and image contents into various languages and formats suited for the users' wireless devices. It does not however, transcode streaming videos. The idea of providing differentiated service to clients of a web server and managing available bandwidth by transcoding JPEG images and dynamically determining the quality of the image to transmit to the client has been explored in [5].

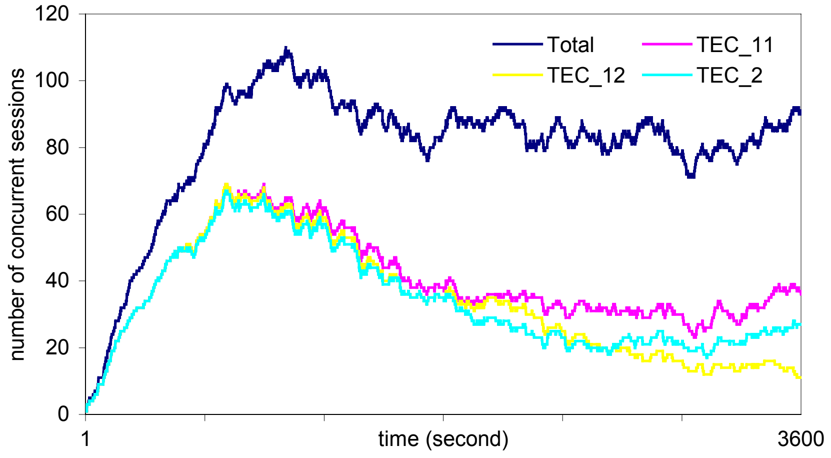
In [16], a layered caching scheme is introduced to adjust stream quality based on per-layer popularity. The video layers are inter-dependent while in our work, different versions of a video object are independent. In [16], the proxy serves the end user with part of the object while fetching other parts (i.e., enhancement layer) if the video gains popularity. The concept of partial hit is thus introduced. However, the transcode hit defined in our work is different; a transcode hit contains a super set of the content that the end user requests. The TeC proxy serves the end user with a transcoded version, and no access to the origin server is required. Similar to [16], [13] proposed a soft caching system for caching Web images at different resolutions. It evaluated the soft caching system for images based on the user download time.



(a)



(b)



(c)

Figure 8. Simulation results for dominant-variant scenario in master-version-origin model.

5 Conclusion

We proposed a transcoding-enabled caching proxy system. Two types of caching algorithms specific to TeC proxy system were presented. In addition to an efficient video delivery to the end users with heterogeneous network conditions and client capabilities, the proposed system improves caching performance by serving transcoded objects to the client and intelligently caching them. Both trace-driven simulation and model-based simulation are conducted. Simulation result shows that our proposed system improves caching performance with marginal transcoding load in the enterprise network environment. For network environment with users of more heterogeneous network conditions, nearly 20% of increase in caching performance is achieved with manageable computation load on the transcoder. For media sites with only one video version, TeC is still beneficial when most clients access the content through low bandwidth channels.

The TeC proxy system can be extended to manage other types of Web content such as images. Similar strategies can be used. For video contents, interval caching or prefix caching can be deployed rather than storing the entire video. Given a distribution of access intervals, our algorithm can be expanded to operate in a similar fashion and comparable advantages are expected. In addition, algorithms can be developed to adaptively select different flavor of the TEC algorithms based on variations in the user access pattern. This extension is an ongoing work.

References

1. S. Acharya and B. Smith, "MiddleMan: A Video Caching Proxy Server," *Proceedings of NOSSDAV 2000,* Chapel Hill, NC, June 2000.
2. E. Amir, S. McCanne and R. Katz, "An Active Service Framework and its Application to Real-time Multimedia Transcoding," *Proceedings of ACM SIGCOMM'98*, Vancouver, Canada, September 1998, pp. 178-189.
3. H. Bharadvaj, A. Joshi and S. Auephanwiriyakul, "An Active Transcoding Proxy to Support Mobile Web Access," *Proceedings of IEEE Symposium on Reliable Distributed Systems*, West Lafayette, IN, October 1998, pp. 118-123.
4. E. Bommaiah, K. Guo, M. Hofmann, and S. Paul, "Design and Implementation of a Caching System for Streaming Media over the Internet," Proceedings of the 6th IEEE Real-Time Technology and Applications Symposium, Washington, DC, June 2000, pp. 111-121.
5. S. Chandra, C. S. Ellis and A. Vahdat, "Differentiated Multimedia Web Services using Quality Aware Transcoding," *Proceedings of the Nineteenth Annual Joint Conference Of The IEEE Computer And Communications Societies (INFOCOM)*, Tel Aviv, Israel, March 2000.
6. L. Cherkasova and M. Gupta, "Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads," *Proceedings of NOSSDAV 2002*, Miami, FL, May 2002.
7. M. Chesire, A. Wolman, G. M. Voelker, and H. M Levy, "Measurement and Analysis of a Streaming-Media Workload," *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, San Francisco, CA, March 2001.
8. CNN.com Video, http://www.cnn.com/video.
9. 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 Communications*, 5(5), August 1998, pp. 10-19.
10. R. Han, *et al*., "Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing," *IEEE Personal Communications*, 5(7), December 1998, pp. 8-17.
11. IBM Websphere, http://www.developer.ibm.com/websphere/index.html.
12. S. Jin and A. Bestavros, "GreedyDual Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams," *Computer Communications,* 24(2), February 2001, pp. 174-183.
13. J. Kangasharju, Y. Kwon, A. Ortega, X. Yang, and K. Ramchandran, "Implementation of Optimized Cache Replenishment Algorithms in a Soft Caching System," *Proceedings of IEEE Second Workshop on Multimedia Signal Processing,* Los Angeles, CA, December 1998, pp. 233-238.
14. W.Y. Ma, B. Shen, and J. Brassil, "Content Service Network, The Architecture and Protocols," *Proceedings of WCW 2001*, Boston, MA, June 2001, pp. 89-107.
15. E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," *Proceedings of the ACM SIGMOD'93 International Conference on Management of Data*, Washington, DC, May 1993, pp. 297-306.
16. R. Rejaie, H. Yu, M. Handely, and D. Estrin, "Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet," *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000, pp. 980-989.
17. S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999, pp. 1310-1319.
18. B. Shen, and S. Roy, "A Very Fast Video Special Resolution Reduction Transcoder," *Proceedings of International Conf. On Acoustics Speech and Signal Processing (ICASSP),* Orlando, FL, May 2002.
19. J. Smith, R. Mohan, and C. Li, "Scalable Multimedia Delivery for Pervasive Computing," *Proceedings of ACM Multimedia*, Orlando, FL, November 1999, pp. 131-140.
20. 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 Transactions on Networking*, 8(4), August 2000, pp. 429-442.