TRANSCODING-ENABLED CACHING PROXY FOR VIDEO DELIVERY IN HETEROGENEOUS NETWORK ENVIRONMENTS

Bo Shen and Sung-Ju Lee
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
{boshen, sjlee}@hpl.hp.com

ABSTRACT

We propose a transcoding-enabled caching system (TeC) along with a new set of caching algorithms for video delivery over the Internet. 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 hit ratio. Simulation results indicate that by incorporating transcoding capability at the network edges, the traffic to the content origin server is reduced 20%.

KEY WORDS: video transcoding, multimedia caching, mobile multimedia.

1. INTRODUCTION

Delivery of multimedia contents over the Internet is always a challenge; encoding, delivery, caching, and processing are all more difficult than those of simple web objects. For example, when CNN posts a news video clip on its Web server, it first has to encode the news clip at several different bit rates (28-56 Kbps for dialup connections and 150+ Kbps for broadband networks [1]) to satisfy end users with different network connection speeds. In addition, the prosperity of wireless network brings more heterogeneous client devices. Traditional caching system treats each client request independently. Usually, popular items are cached at a proxy close to the end user and therefore, reducing the traffic between the content origin and proxies as well as the user perceived startup latency. However, various versions with different bit rates of the same video clip may be cached at one instance, which can be a waste of storage.

Our solution is to enable the caching proxy with the transcoding capability so that variants of a video object can be delivered with transcoding what is cached in the proxy to the appropriate format, instead of accessing the object from the content origin that may be far away from the client. This approach puts transcoding units in the content delivery path. Putting computing resources in the content delivery path has been addressed in [2,3]. Dynamic content adaptation can be performed by these network intermediaries. One of the advantages of having transcoding-enabled proxy is that the content origin servers may not need to generate different bit rate videos. Moreover, heterogeneous clients with various network conditions will receive videos that are suited for their capabilities, as content adaptation can easily be done at the network edges.

We investigate the possibility of using computing resource to trade off caching storage space, 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. 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 work targets at improving the efficiency of the task. In [4], we were able to transcode DVD video at high resolution to low bit rate video at lower resolution in real time. The results are video objects deliverable to mobile clients through low bandwidth channels.

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. Preliminary results indicate 20% of increase in caching performance with manageable computation load on the transcoders.

The rest of the paper is organized as follows. In Section 2, basic background for transcoding and the architecture of the transcoding-enabled caching proxy are introduced. In Section 3, we propose a set of caching algorithms for transcoding-enabled proxies. Performance analysis and simulation results are presented in Section 4. Related work is discussed in Section 5. We conclude in Section 6.

2. SYSTEM ARCHITECTURE

Caching proxy is often deployed at the edges of the network to reduce the traffic to the origin server and user perceived latency. We propose a transcoding enabled caching proxy that serves variants of objects to the end users with different devices or connection profiles. Focusing on the video delivery, we illustrate the system architecture as follows.

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

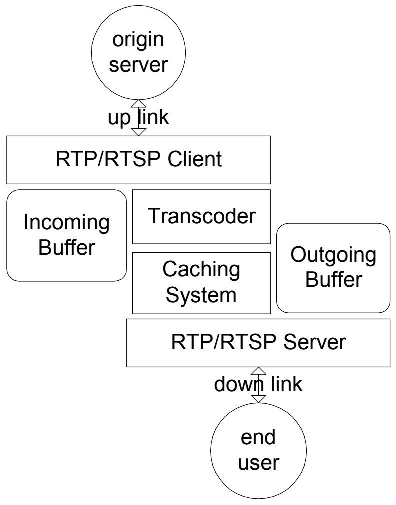


Figure 1 System and components for transcoding-enabled caching proxy.

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 generated by the transcoder per second. As long as the transcoding bit rate is larger than the minimum of the bandwidth of the uplink and the downlink, the transcoding process does not significantly increase the end-to-end delay. However, video transcoding can be computing intensive. Many works are under investigation to reduce the workload of such a session. Among them, compressed domain based approach provides the best performance [4]. In compressed domain transcoding, the incoming video is only partially decompressed and rate adapting is performed in the compressed domain while the motion information is reused. This approach significantly 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 brought by the collocated transcoder.

Given the real time transcoding capability, the TeC proxy can dynamically transcode objects to different variants to satisfy the end users with various capabilities. 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 version at lower bit rate. For example, a video at bit rate of 64 Kbps can be transcoded from the same video at bit rate of 128 Kbps, and the 128 Kbps version is a transcodable version for the one at 64 Kbps. 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.

3. CACHING STRATEGIES

A TeC proxy trades off computation with storage. The main idea is to serve the end user with the transcoded version of what is available in the cache whenever possible. 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., $b_1 > b_2 > … > b_n$. 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) available in cache, the TeC proxy will transcode 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 requested object exists in the cache.
- *Transcode Hit*, the requested version does not exist in the cache, but a transcodable version of the requested object does.
- Miss, the requested version of the requested object nor a transcodable one does not exist in the cache.

Depending on the caching behavior when each event occurs, we propose two types of caching algorithms.

3.1 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 time. By caching only one version, we are utilizing the storage more efficiently and caching more video objects. The main challenge of this algorithm is to decide which bit-rate version of the video to cache.

When there is an exact hit, the TeC proxy refreshes the access record for the hit. Then the requested version of the video object is streamed 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. However, remember that we are considering each version of a video as an item; even if a request is received to a video that is stored in the cache, if that request had 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. Since we only allow one version of an object to exist in the cache, the lower bit rate version is evicted. In general, if the end user requests version b_i of a video object while b_j, where $b_i > b_j$, exists in the cache, then b_j is evicted before b_i is fetched from origin server and subsequently cached at the proxy.

If an access results in a transcode hit, the TeC proxy transcodes it to an appropriate version and streams the transcoded version to the end user. In the mean time, the proxy can choose to cache in two different ways, which leads to two variations of the algorithm. For algorithm TEC_11, the proxy chooses to refresh the access record of the transcode hit without caching the newly transcoded version. For algorithm TEC_12, the proxy chooses to evict the transcode hit and caches the newly transcoded version. In general, if the end user requests version b_i of a video object while b_j, where $b_i < b_j$, exists in the cache, then b_i is transcoded from b_j and streamed to the end user. For TEC_11, the access record of b_j is refreshed. For TEC_12, b_j is first evicted and b_i is cached.

Whenever the cache becomes full and requests to the video that are not in the cache are received, certain files in the cache must be replaced. We can simply use the existing popular cache replacement algorithms (e.g., LRU, LFU, LRU-*k* [6], or GD* [7]).

3.2 Cache Multiple Versions (TEC_2)

The motivation of caching multiple versions of the same object is to reduce the 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 performance.

In this algorithm, when there is a cache miss, the TeC proxy fetches the requested version, streams it to the end user and caches it even though there may be other versions of the same video object in the cache. Effectively, multiple popular versions of a popular video can be cached at a given time.

If a transcode hit occurs, the transcoder is invoked to generate the particular version requested. 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, then b_i is transcoded from b_j, delivered to the end user and subsequently cached. Note that in this case, b_i and b_j both exist in the cache.

Similarly, when the cache becomes full, we use an existing algorithm for the replacement of each item. For example, if LRU algorithm is used, TeC proxy will find the least recently used objects and replace them with either the fetched or the transcoded version.

4. SIMULATION MODEL AND RESULT

There are limited video traces available to the public, and to make things worse, even less for variant-based web traces. We choose to simulate access pattern to the best of our knowledge by setting the parameters found in [7]. In all the simulations, LRU is used for cache replacement algorithm and a stack-based implementation is developed.

4.1 Simulation Parameters

To evaluate the performance of the proposed algorithms, we simulate the video access patterns as follows. Given a pool of 500 original video objects of lengths varying from one minute to five minutes, the access pattern is characterized by 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 1 Mbps and 512 Kbps for desktop users, and 256 Kbps for mobile users.

Using the above parameters, we generate a content pool of about 20 GB of data. Based upon various cache capacities, we study the byte hit ratio of transcoding-enabled caching proxy.

4.2 Reference Scenario

Our algorithms are compared with the case where the caching proxy is not transcoding-enabled. That is, the original content server stores all different versions of each original video object. The 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 video object but a different version. We call this algorithm a reference model.

4.3 Simulation Scenario

For TeC simulation, the content origin server stores all different versions for each video. The TeC proxy serves transcoded versions based on requests from the end users and the available versions in the cache storage.

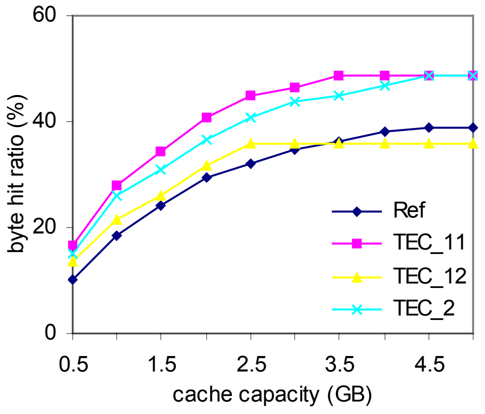
We define a "dominant-variant" scenario as follows. In this scenario, we picture the requests are mainly from mobile users after work checking news through cell phones or PDAs. The access arrival interval is short and highly temporally localized. We simulate this scenario by a one-hour simulation. In this experiment, 600 accesses arrive through a random Poisson process. Therefore, the average access arrival interval is six seconds. In this case, the variation in downlink capacity is relatively small. We simulate this by assigning 80% of the downlink traffic to dominant downlink capacity. For example, we assume that 256 Kbps downlink from mobile device is the dominant link while there are occasional requests from 512 Kbps or 1 Mbps downlink from desktops.

We further define an "even-variant" scenario as follows. In this scenario, video contents are accessed evenly through all three types of clients. This scenario represents a highly heterogeneous environment. While the popularity of video objects still follows a Zipf-like distribution (α=0.47), the downlink capacity varies evenly. The simulation time is extended to 12 hours with 600 accesses arriving through a random Poisson process. Effectively, the average access arrival time interval is extended to 72 seconds.

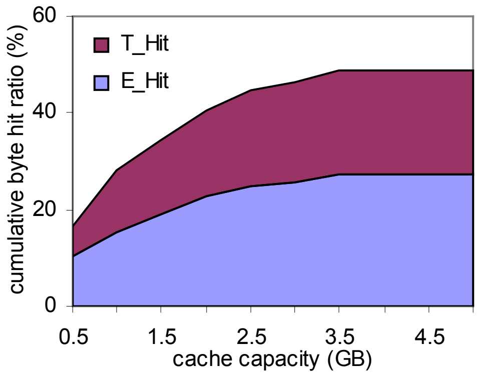
4.4 Simulation Result

We use the byte hit rate to evaluate each algorithm under each simulation scenario. Note that the byte hit ratio represents what percentage of the bytes of the content the end users requested are replied directly from the TeC proxy. The larger the byte hit ratio, the better the performance.

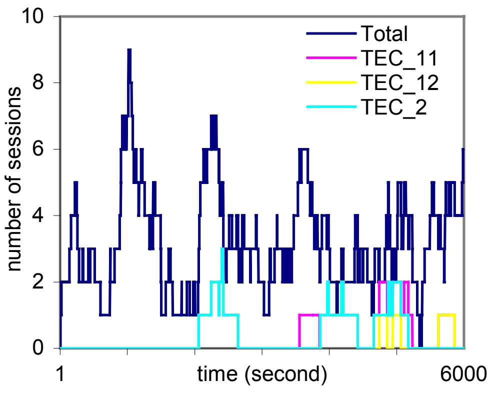
Figure 2 (a) shows the byte hit ratio for various cache capacities in even-variant scenarios. Note that TEC_11 algorithm provides the best result, which is nearly 30% better than the reference model. Figure 2 (b) shows the portion of exact hit (E_Hit) and transcode hit (T_Hit) for TEC_11 algorithm. The result reflects the fact that 20% to 50% of the byte can be served through transcoding. Figure 2 (c) shows the total number of streaming sessions and the number of transcoding sessions for each caching algorithm when the cache capacity is 2 GB. The result for the first 6000 seconds of the 12-hour simulation is shown. The number of concurrent transcoding sessions is well within the range that a current PC can cope with.



(a)

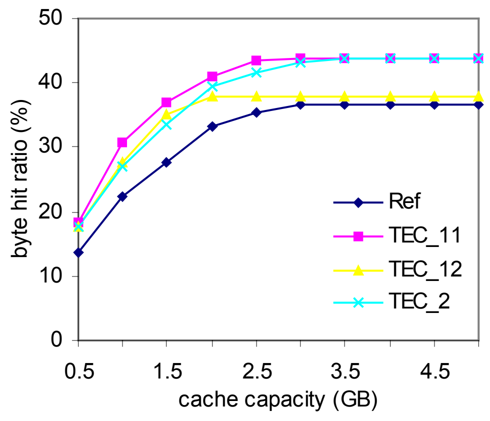


(b)

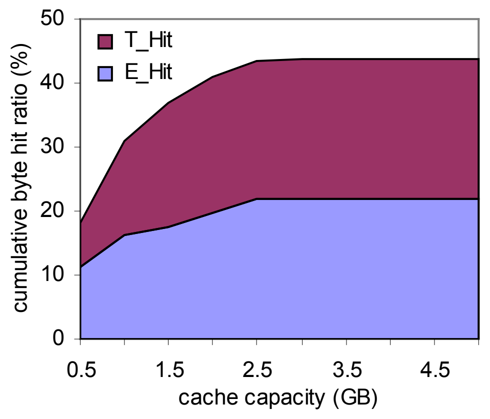


(c)

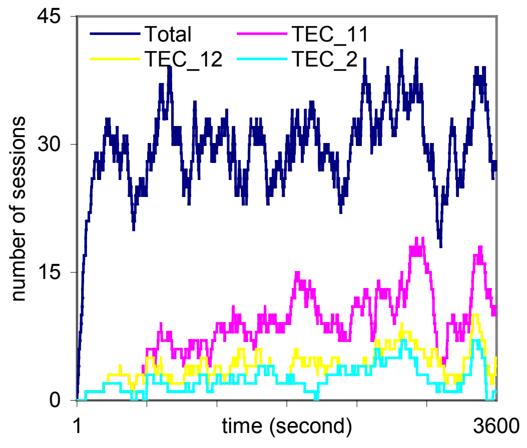
Figure 2 Simulation result for even-variant scenario.



(a)



(b)



(c)

Figure 3 Simulation results for dominant-variant scenario.

For dominant-variant scenario, Figure 3 shows similar results except that the benefit gained from the TEC algorithms is reduced compared to the even-variant scenario.

Note that from Figure 3 (c), the number of total streaming sessions and transcoding sessions is increased significantly comparing with Figure 2 (c). This is due to the high density of client accesses. For TEC_11, up to 50% of the total session needs transcoding, which poses a heavy load on the transcoding unit. Variations of the TEC algorithms provide different level of load on the transcoder. TEC_11 has the highest level of transcoding load, while TEC_2 has the lowest. This is not surprising since the TEC_11 algorithm tries to cache the highest bit rate version, while TEC_2 tries to cache multiple versions and therefore an exact hit is more likely to occur especially when accesses are highly temporally localized. In practice, one can choose to use different TEC algorithms based on the transcoding resource available. For example, if the TeC proxy can handle at most ten concurrent transcoding sessions, TEC_2 is a better choice since it incurs at most eight concurrent transcoding sessions according to Figure 3 (c). On the other hand, TEC_2 still produces around 20% increase in byte hit ratio comparing with the reference model.

5. RELATED WORK

Several schemes for caching video streams from the Internet have been proposed. Prefix caching [8] stores the initial parts of popular videos on the proxy to reduce the playback latency. MiddleMan [9] aggregates a number of proxy caches connected within a network to cache video files. The system proposed in [10] has proxies perform request aggregation, prefix caching, and rate control to cache streaming media. Video staging [11] prefetches certain videos onto the proxies to preserve WAN bandwidth. All of the above schemes do not use transcoding at the local network proxies.

Many works have contemplated with the idea of putting transcoders at network intermediaries [12-16]. MOWSER [12] allows mobile users to specify the QoS parameters. Proxy agents between the web server and mobile users transcode the web content into viewing preference to the clients. MOWSER however, does not deal with proxy caching. A similar work is done at [13] 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 [14]. 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 [15]. IBM Websphere transcoding publisher [16] can transcode HTML and image contents into various languages and formats suited for the users' wireless devices. It is not however, capable of transcoding streaming objects.

In [17], a layered caching scheme is introduced to adjust stream quality based on per-layer popularity. The video layers are dependent. In our work on the other hand, different versions of a video object are independent. In [17] the proxy can serve the end user with part of the object while fetching for other parts (enhancement layer) if the video gets popular. The concept of partial hit is thus introduced. However, the transcode hit defined in our work is different. A transcode hit contains a supper 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 server is required.

Similar to [17], [18] proposed a soft caching system for caching Web images at different resolutions. It is similar to our work in the sense that different resolutions of the same image can be cached at the proxy. However, the tradeoff between computation and storage was not considered. In addition, [18] evaluated the soft caching system for images based on the user download time. Video caching poses a different set of challenges. Our work investigates the benefits transcoder brings to caching. Download time is less of a factor given that the video is streamed from end to end.

6. CONCLUSION

We proposed a transcoding-enabled caching proxy system. Two types of caching algorithms specific to TeC proxy system are 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. Simulation result indicates 20% of increase in caching performance with manageable computation load on the transcoder.

The TeC proxy system can be extended to deal with other types of Web contents such as images. Similar strategies can be used. For video contents, interval 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 can be expected.

REFERENCES

[1] http://www.cnn.com/video

[2] W.Y. Ma, B. Shen, J. Brassil, Content Service Network, The Architecture and Protocols, *Proc. 6th Internation Web Content Caching and Distribution Workshop*, Boston, MA, 2001, 89-107.

[3] E. Amir, S. McCanne and R. Katz, An Active Service Framework and its Application to Real-time Multimedia Transcoding, *Proc. SIGCOMM'98*, Vancouver, B.C. 1998, 178-189.

[4] B. Shen, S. Roy, A Very Fast Video Special Resolution Reduction Transcoder", *Proc. International Conf. On Acoustics Speech and Signal Processing (ICASSP)*, May 2002.

[5] 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, 297-306.

[6] S. Jin and A. Bestavros, "GreedyDual Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams," *Computer Communications*, *24*(2), Feb. 2001, 174-183.

[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 (USITS'01)*, San Francisco, CA, March 2001.

[8] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'99)*, New York, NY, March 1999, 1310-1319.

[9] S. Acharya and B. Smith, "MiddleMan: A Video Caching Proxy Server," *Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Video and Audio (NOSSDAV 2000)*, Chapel Hill, NC, June 2000.

[10] 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 (RTAS 2000)*, Washington, DC, June 2000, 111-121.

[11] 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, 429-442.

[12] H. Bharadvaj, A. Joshi and S. Auephanwiriyakul, An active transcoding proxy to support mobile web access, *Proc. 17th IEEE Symposium on Reliable Distributed Systems*, 1998, 118-123.

[13] J. Smith, R. Mohan, and C. Li, Scalable multimedia delivery for pervasive computing, *Proc. ACM Multimedia*, Orlando, FL, 1999, 131-140.

[14] A. Fox, S. D. Cribble, Y. Chawathe & E. A. Brewer, Adapting to network and client variation using infrastructural proxies: lessons and perspectives, *IEEE Personal communications, 5*(5), Aug. 1998, 10-19.

[15] R. Han, P. Bhagwat, et al., Dynamic adaptation in an image transcoding proxy for mobile web browsing, *IEEE Personal communications, 5*(7), Dec. 1998, 8-17.

[16] IBM Websphere, http://www.developer.ibm.com/websphere/index.html

[17] R. Rejaie, H. Yu, M. Handely, D. Estrin, Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet, *Proc. of INFOCOM*, Tel Aviv, Israel, Mar. 2000, 980-989.

[18] J. Kangasharju, Y. Kwon, A. Ortega, X. Yang, K. Ramchandran, Implementation of Optimized Cache Replenishment Algorithms in a Soft Caching System, Proc. *1998 IEEE Second Workshop on Multimedia Signal Processing*, Dec. 1998, 233-238.