

Quality-Adaptive Proxy Caching for Peer-to-Peer Video Streaming Using Multiple Description Coding^{*}

JEN-WEN DING AND SHI-YUAN LAN
Department of Information Management
National Kaohsiung University of Applied Sciences
Kaohsiung, 807 Taiwan
E-mail: jwding@cc.kuas.edu.tw

In the past few years, video streaming has been one of the most prominent applications in the Internet. There are basically two types of approaches to deliver on-demand video streams: infrastructure-based and infrastructure-less. The infrastructure-based approach, also known as the content delivery network (CDN) approach, employs numerous replicate servers and proxies to assist video streaming. The main drawback of this approach is that it is costly to deploy numerous servers and proxies. The infrastructure-less approach, also known as the peer-to-peer (P2P) approach, employs numerous peers to relay video streams. The main drawback of the P2P approach is its instability due to frequent joining/leaving of peers. The state-of-the-art development is to combine the two types of approaches. In this paper, we employ multiple description coding (MDC) to develop a novel P2P-assisted proxy caching scheme, called MP³, for on-demand video streaming. Our simulation results show that with the use of MDC, MP³ can effectively provide adaptive video quality for clients in a heterogeneous network environment.

Keywords: adaptive video streaming, peer-to-peer computing, multimedia proxy, cache replacement algorithm, multiple description coding

1. INTRODUCTION

With the rapid development of the Internet and multimedia technology, real-time video streaming has become a very popular service. There are two main challenges concerning the delivery of real-time streams. First, because the size of video data is very large, real-time streaming usually imposes a heavy load on servers [6, 14]. Secondly, real-time streaming imposes both a heavy load and a strict QoS requirement on networks. Much research has been devoted to delivering real-time streams over wired/wireless networks [3-5, 7, 13, 15]. Real-time streaming can be divided into two types of categories: live streaming and on-demand streaming. Live streaming is usually employed to broadcast an ongoing event, such as news, sports, weather, or a TV channel. On-demand streaming is usually used to deliver personalized content, such as movies and dramas. Live streaming is much simpler than on-demand streaming since the users of live streaming request the same content at the same time. Therefore, live streaming is usually implemented by network-layer or application-layer multicast or broadcast. On the other hand, on-demand streaming is more challenging since each user may request different content at the same time, which imposes a much heavier load on both the servers and the underlying networks.

Received February 1, 2008; accepted November 28, 2008.

Communicated by Yau-Hwang Kuo, Pau-Choo Chung and Jar-Ferr Yang.

^{*}This work was partially supported by the National Science Council of Taiwan, R.O.C., under research grant No. NSC-97-2221-E-151-012.

Previous research has shown that two types of delivery approaches can effectively reduce the server/network loads imposed by on-demand streaming: infrastructure-based approach and infrastructure-less approach. The infrastructure-based approach, also known as the content delivery network (CDN) approach, employs numerous replicate servers and proxies to assist video streaming. The main disadvantage of this approach is that it is expensive to deploy numerous servers and proxies. The infrastructure-less approach, also known as the peer-to-peer (P2P) approach, employs users (peers) to relay video streams. The main disadvantage of the P2P approach is its unstableness due to the frequent joining/leaving of peers. The state-of-the-art development for on-demand video streaming is to combine both types of approaches, such as the PROP system [11], COPACC system [16], and the scheme proposed in [20, 26]. Besides the two types of approaches, it is worth noting that in wireless networks, broadcasting techniques is usually employed to deliver hot on-demand streams [8, 18, 19, 28].

However, in most of the combined approaches, it is assumed that all clients request the same video quality, which, unfortunately, is an oversimplified assumption in practice, especially in a heterogeneous network environment. To deal with the client heterogeneity problem, scalable video coding is usually employed. Scalable video coding can be classified into three categories: cumulative layered coding, fine-grained scalability (FGS) coding, and multiple description coding (MDC). In cumulative layered coding, a video file is encoded into one base substream (also called layer) and one or many enhancement substreams. Each enhancement substream must be completely received in order to be successfully decoded. In FGS coding, a video stream is encoded into one base substream and one enhancement substream. The salient feature of FGS coding is that the enhancement substream can be successfully decoded without being completely received. In MDC coding, a video stream is encoded into several independent substreams [10]. The main feature of MDC coding is that each substream can be successfully decoded without the existence of the base layer. In this paper, we do not consider cumulative layered coding and FGS coding since both coding methods require the clients to have the base layer. However, it is possible that the peers holding the base layer quit the system due to the frequent joining/leaving behavior of peers. This phenomenon, unfortunately, makes other peers difficult to decode the required video even though they have received one or many enhancement substreams. In this paper, we employ MDC to develop a novel caching scheme for P2P-based video proxy, called MP³ (MDC-based P2P video Proxy). The main feature of MP³ caching scheme is that it can provide video streams with adaptive quality for heterogeneous clients.

The rest of this paper is organized as follows. Section 2 provides an overview of the recent development of video proxies. Section 3 describes our proposed scheme, MP³. Section 4 presents the performance evaluation results of MP³. Finally, section 5 concludes this paper.

2. PREVIOUS WORK

Over the past decade, there have been many studies proposed to reduce server/network loads caused by on-demand streaming. In this section, we briefly review recent related works in the literature. Three types of delivery approaches are reviewed: proxy-based approach, P2P-based approach, and the hybrid approach.

2.1 Proxy-Based Approach

Recent works on the proxy-based approach take advantage of scalable video coding or transcoding to develop caching mechanisms for adaptive on-demand streaming. In [29], Zhang *et al.* developed a cost-based caching algorithm for single server and multiple servers. Their work employs cumulative layered video, and their cost-based caching algorithm considers multiple objectives, such as video quality, network throughput, network utilization, and client startup latency. In [21], Liu *et al.* developed a novel adaptive video caching framework that enables low-cost and fine-grained adaptation. Their work employs MPEG-4 FGS video with post-encoding rate control. With their caching algorithm, heterogeneous streaming rates can be supported and the backbone bandwidth consumption can be adaptively controlled. In [17], Kao and Lee employed transcoding to design a novel proxy caching algorithm. The proposed scheme can provide multi-rate streaming services for heterogeneous clients. The main idea of their work is that when a client requests a simplified version of a video object and the proxy has a more detailed version of the object, then the proxy will perform transcoding without forwarding the request to the server, thereby reducing the load imposed on both the servers and networks.

2.2 P2P-Based Approach

In the past few years, some researchers start to investigate how to employ P2P relay to assist on-demand video streaming. Compared to the proxy-based approach, the deployment cost of the P2P-based approach is very low. Well-known P2P approaches include P²Cast [12] and P2VoD [9]. To deal with the client heterogeneity problem, Liu *et al.* [22] develops a P2P-based low-cost video-on-demand system for heterogeneous receivers using category overlay architecture. However, the P2P-based approach is intrinsically unstable due to the frequent joining/leaving of peers.

2.3 Hybrid Approach

The hybrid approach is to combine video proxies and P2P relay to assist on-demand streaming. A few hybrid approaches have been proposed. Two prominent schemes are briefly reviewed. In 2006, Guo *et al.* proposed a scalable and reliable P2P assisted proxy architecture and developed a new caching algorithm [11]. In their work, an ideal redundant cache model is first developed. The cache model can find out the optimal distribution of video segments, *i.e.*, the optimal number of replicas of each video segment that should be cached in the system in order to minimize the overall missed requests (*i.e.*, the requests that cannot obtain the required data segment in the system). However, in practice, it is difficult to carry out the ideal model. Therefore, they present a heuristic distributed algorithm, called PROP, which employs a utility function to approach the optimal distribution. In 2007, Ip *et al.* propose a COPACC system, which employs a P2P-based multi-proxy architecture [16]. COPACC can effectively reduce the communication costs of on-demand streaming. In their approach, each video file is divided into three parts: prefix, prefix-of-suffix, and suffix. The prefix part is stored in the proxies; the prefix-of-suffix part is stored in the clients; and the remaining part is directly retrieved from the

server. The main feature of COPACC system is that it can derive the optimal partition of each video (*i.e.*, the optimal size of the prefix part and prefix-of-suffix part) and can obtain the optimal allocation of prefix and prefix-of-suffix. However, their work does not consider the problem of heterogeneous clients. Neither PROP nor COPACC considers the use of scalable video coding, which, however, is an important design issue for adaptive video streaming in heterogeneous network environments.

3. MP³ CACHING ALGORITHMS

3.1 System Overview

Fig. 1 shows the main components of our system architecture: video server (or server cluster), video proxy, and video clients (peers). A client plays three roles. First, it is a requester, sending requests to other peers or the proxy. Secondly, it is a provider, sending cached video object to other requesters. Thirdly, it is also an index server, helping to relay requests to other peers and keeping the access information of the cached video objects used by the MP³ caching algorithms.

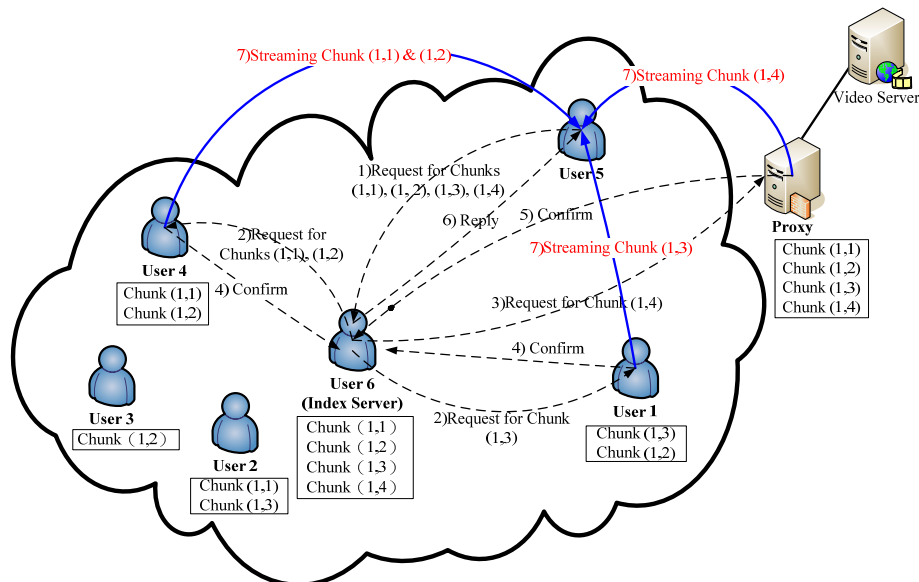


Fig. 1. Architecture and operation of the MP³ system.

In MP³, the basic data unit for caching is called “data chunk”. A data chunk is a video segment of an MDC description, as shown in Fig. 2. The size of a video segment is an adjustable parameter. All clients can make use of distributed hash table (DHT) to search the required data chunks among peers using some DHT schemes, such as CAN and Chord [23]. DHT supports three main functions, *Put(key, data)*, *Get(key)*, and *Remove(key)*, where “data” represents the data object to be stored and retrieved among peers and “key” represents the identifiers of the data objects. In this paper, data objects refer

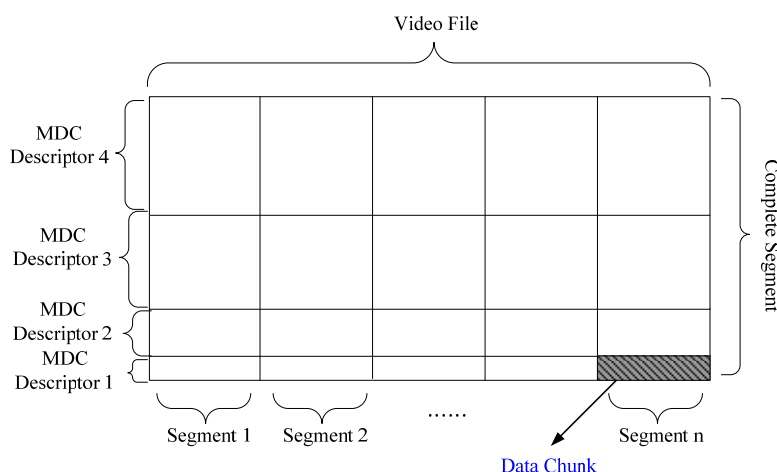


Fig. 2. Definition of a data chunk.

to video data chunks. A key of a data chunk is the hashed value of its video ID and chunk ID. A peer can make use of *Put(key, data)* to declare that it has cached a new data chunk, *Get(key)* to find a required data chunk, and *Remove(key)* to delete a data chunk from its caching space. To make the streaming more efficiently, the network locations of peers can be embedded in the keys. By doing so, peers can use DHT to find out geographically closer peers for efficient streaming. For locality-aware DHT-based P2P streaming, interested readers can refer to [27].

When a user needs new data chunks, it first checks other peers, and then the proxy, and finally the server. Specifically, at first, a user will issue a request, via DHT, to other peers who have cached the requested data chunks. If the first inquired peer is incapable of serving the request, it then serves as an index server. The index server will forward the request to other peers who have the requested data chunks. If there exist some peers who are able to serve the request, those peers will respond to the index server. If no peers are able to serve the request, the request will be sent to the video proxy. If the video proxy does not keep the required data chunk, then the proxy will forward the request to the video server. Whenever a data chunk is requested, the index server will update the access information of the requested data chunks. Also, the index server will notify all peers holding the same data chunks to perform a similar update (each peer needs to maintain some access information for the data chunks it caches, as discussed in section 3.3).

In what follows, we use an example shown in Fig. 1 to explain the operation of the caching process discussed above.

- (1) A new user, say user 5, issues a request, via DHT, to user 6 to request four data chunks, chunk(1, 1), chunk(1, 2), chunk(1, 3), and chunk(1, 4), where $Chunk(i, j)$ represents segment i of description j . If user 6 is incapable of serving the request (e.g., it may have insufficient bandwidth for streaming the required data chunks to peer 5), then user 6 serves as an index server and forwards the request to other peers holding the requested data chunks.

- (2) User 6 sends a request to user 4, who has chunk(1, 1) and chunk(1, 2). It also sends a request to user 1, who has chunk(1, 2) and chunk(1, 3).
- (3) If user 6 cannot find any peer who has chunk(1, 4), it will send a request to the proxy, asking for chunk(1, 4).
- (4) If user 1 and user 4 are able to serve the requests, they will send a confirmation message to user 6.
- (5) If the proxy has chunk(1, 4) and is able to stream the data chunk, it will send a confirmation message to user 6.
- (6) When user 6 receives the confirmation message from user 1, user 4, and the proxy, it will notify user 5 of how to get the required data chunks.
- (7) Finally, user 5 will request the required data chunks directly from user 1, user 4, and the proxy, respectively. User 1, user 4, and the proxy then directly stream the requested data chunks to user 5. At the same time, user 6 will update the access information regarding the four data chunks and notifies all peers caching the same data chunks to perform a similar update.

3.2 Caching Principles

In MP³ caching algorithms, multiple design objectives are considered at the same time. This subsection explains the main caching objectives for both the proxy and clients (peers).

There are four design principles for caching data chunks in a video proxy.

- (1) A proxy assigns higher caching priority to the data chunks with higher popularity, except for the data chunks with too many replicas in the system, to provide stable streaming to popular data chunks.
- (2) A proxy assigns higher caching priority to the data chunks with higher data consumption rate, thereby reducing the traffic between the video server and proxy.
- (3) A proxy assigns higher caching priority to the data chunks with higher utility, which will be defined later.
- (4) A proxy assigns higher caching priority to the data chunks belonging to the initial part of a video, thereby reducing users' start-up service latency.

Similarly, there are four design principles for caching data chunks in peers.

- (1) A peer assigns higher caching priority to the data chunks with higher utility, which will be defined later.
- (2) A peer assigns higher caching priority to the data chunks with higher quality reward functions, which will be defined later.
- (3) A peer assigns higher caching priority to the data chunks with fewer replicas in the system, thereby avoiding the over-proliferation of hot data chunks.
- (4) A peer assigns higher caching priority to the data chunks belonging to the initial part of a video file, thereby reducing users' start-up service latency and reducing the traffic between video servers and proxies. Note that by doing so, the initial parts of unpopular videos also have chance to be cached by peers.

Table 1. Parameters used by MP³ system.

Parameters	Definition
$\delta_{i,j}$	Chunk(i, j), <i>i.e.</i> , segment i of description j
$p_{i,j}$	Popularity of chunk $\delta_{i,j}$
$T_{i,j}^*$	Last access time of chunk $\delta_{i,j}$
$T_{i,j}^0$	First access time of chunk $\delta_{i,j}$
T_{now}	Current time
R_T^{pxy}	Transmission reward function used by proxy
R_U^{pxy}	Utility reward function used by proxy
R_L^{pxy}	Latency reward function used by proxy
$W_T^{pxy}, W_U^{pxy}, W_L^{pxy}$	Weights of $R_T^{pxy}, R_U^{pxy}, R_L^{pxy}$
$R_{T,max}^{pxy}, R_{U,max}^{pxy}, R_{L,max}^{pxy}$	Maximum value of $R_T^{pxy}, R_U^{pxy}, R_L^{pxy}$
R_{sum}^{pxy}	Weighted sum of $R_T^{pxy}, R_U^{pxy}, R_L^{pxy}$
γ	Anti-proliferation parameter used by proxy
R_L^{peer}	Latency reward function used by peers
R_U^{peer}	Utility reward function used by peers
R_Q^{peer}	Quality reward function used by peers
M	Number of MDC descriptions
$W_L^{peer}, W_U^{peer}, W_Q^{peer}$	Weights of $R_L^{peer}, R_U^{peer}, R_Q^{peer}$
$R_{L,max}^{peer}, R_{U,max}^{peer}, R_{Q,max}^{peer}$	Maximum value of $R_L^{peer}, R_U^{peer}, R_Q^{peer}$
R_{sum}^{peer}	Weighted sum of $R_L^{peer}, R_U^{peer}, R_Q^{peer}$
β	Anti-proliferation parameter used by peers
$N_{i,j}^{rep}$	Number of replicas of chunk $\delta_{i,j}$
$N_{i,j}^{acc}$	Number of access times of chunk $\delta_{i,j}$
$PSNR(\delta_{i,j})$	PSNR value of chunk $\delta_{i,j}$
$Size(\delta_{i,j})$	Size of chunk $\delta_{i,j}$

In what follows, we describe the caching algorithms for both proxies and peers based on the above design principles. The parameters used in the caching algorithms are summarized in Table 1.

3.3 Proxy Caching Algorithm

Let $\delta_{i,j}$ denote chunk (i, j) of video v , *i.e.*, segment i of description j of video v . For ease of exposition, we remove the superscript v in the following. However, it is easy to extend the following model with superscript v . In section 4, our simulation result is based on the extended model with superscript v . To estimate the popularity of a data chunk, say $\delta_{i,j}$, we adopt the estimation method used in [2, 11] and makes it suitable for MDC-based videos. The popularity of chunk $\delta_{i,j}$ can be computed by Eq. (1). The first term $N_{i,j}^{acc} / T_{i,j}^* - T_{i,j}^0$ represents the average access frequency of data chunk $\delta_{i,j}$. The second term is a tendency function, which indicates the latest tendency for the change of the access frequency.

$$p_{i,j} = \frac{N_{i,j}^{acc}}{T_{i,j}^* - T_{i,j}^0} \times \min \left(1, \frac{(T_{i,j}^* - T_{i,j}^0)/N_{i,j}^{acc}}{T_{now} - T_{i,j}^*} \right) \quad (1)$$

Note that $p_{i,j}$ is an estimated parameter, which represents the future access popularity of data chunk $\delta_{i,j}$. To reduce the network traffic between the server and the proxy, we define a transmission reward function in Eq. (2). Note that the transmission reward function of a data chunk is high if the chunk has large size and high access popularity, which represents a heavy load on the network.

$$R_T^{pxy}(\delta_{i,j}) = Size(\delta_{i,j}) \times p_{i,j} \quad (2)$$

To make efficient use of caching space, we define a utility reward function in Eq. (3). A data chunk has high utility if it is accessed frequently with small cached size.

$$R_U^{pxy}(\delta_{i,j}) = N_{i,j}^{acc} / Size(\delta_{i,j}) \quad (3)$$

Note that there exists occasional inconsistency between the two reward functions defined in Eqs. (3) and (2). The inconsistency occurs in two situations. The first situation occurs when a data chunk has low transmission cost reward and high utility reward. The second situation occurs when a data chunk has a high transmission cost reward and a low utility reward. Both situations occur for only videos with low popularities. This inconsistency is primarily due to the intrinsic inconsistency of the two caching goals.

To reduce users' initial waiting time, we take into account the relative location of a data chunk within a video file, and define a latency reward function in Eq. (4), where prefix means the first segment.

$$R_L^{pxy}(\delta_{i,j}) = \begin{cases} e^{p_{i,j}}, & \text{if } \delta_{i,j} \in \text{prefix} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Finally, in Eq. (5), we integrate the three reward functions. When the caching space of the proxy is full, the proxy will use Eq. (5) to evaluate the importance of all data chunks. The data chunk with the lowest importance will be replaced out. It can be observed from Eq. (5) that each reward function is assigned a weight. These weights are adjustable parameters, which can be determined by system designers to achieve various design tradeoffs. In addition, to avoid any reward function dominates the other two, each function is normalized such that the output value of each one ranges from zero to one. Finally, the sum of the three reward functions is penalized by the number of replicas of the data chunk. A parameter, called anti-proliferation parameter γ , is used to limit the over-proliferation of hot data chunks. That is, when a data chunk is already cached by a large number of peers, the caching priority for the data chunk in the proxy should be lowered to make more efficient use of the proxy space. It is worth noticing that when the anti-proliferation parameter γ is high, hot data chunks tend to be replaced out from the proxy, which may result in frequent caching replacement. On the contrary, when the value of γ is low, hot data chunks tend to be kept in the proxy, which may result in inefficient use of the caching space of the proxy. In our experience, $\gamma = 1$ results in good system performance.

$$R_{sum}^{pxy}(\delta_{i,j}) = \frac{W_T \times \frac{R_F^{pxy}(\delta_{i,j})}{R_{F,max}^{pxy}} + W_U \times \frac{R_U^{pxy}(\delta_{i,j})}{R_{U,max}^{pxy}} + W_L \times \frac{R_L^{pxy}(\delta_{i,j})}{R_{L,max}^{pxy}}}{(N_{i,j}^{rep})^\gamma} \quad (5)$$

3.4 Peer Caching Algorithm

To reduce users' initial waiting time, we also define a latency reward function for peers as follows.

$$R_L^{peer}(\delta_{i,j}) = \begin{cases} e^{p_{i,j}}, & \text{if } \delta_{i,j} \in \text{prefix} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

The utility reward function for peers is defined in Eq. (7). If a data chunk is accessed frequently, it will be assigned a higher utility reward function. The cache space occupied by a data chunk is not considered here because the aggregate caching space of all peers is huge, and thus the cache space plays less important role in the peer caching algorithm.

$$R_U^{peer}(\delta_{i,j}) = N_{i,j}^{acc} \quad (7)$$

To obtain the best caching gain from the point of view of video quality, we define a quality reward function in Eq. (8). A data chunk with higher visual quality (in terms of PSNR), smaller size, and higher popularity will be assigned a higher quality reward. The PSNR of a data chunk can be calculated in advance [24].

$$R_Q^{peer}(\delta_{i,j}) = \frac{PSNR(\delta_{i,j})}{Size(\delta_{i,j})} \times p_{i,j} \quad (8)$$

Eq. (9) integrates the above three reward functions. When the caching space of a peer is full, the peer can use Eq. (9) to evaluate the importance of all data chunks to be cached. The data chunks with the lowest importance will be replaced out. Not that, an anti-proliferation parameter, β , is used, which plays a similar role as α . In our experience, $\beta = 1$ yields good system performance.

$$R_{sum}^{peer}(\delta_{i,j}) = \frac{W_Q^{peer} \times \frac{R_Q^{peer}(\delta_{i,j})}{R_{Q,max}^{peer}} + W_U^{peer} \times \frac{R_U^{peer}(\delta_{i,j})}{R_{U,max}^{peer}} + W_L^{peer} \times \frac{R_L^{peer}(\delta_{i,j})}{R_{L,max}^{peer}}}{(N_{i,j}^{rep})^\beta} \quad (9)$$

3.5 Error Recovery

MP³ caching scheme can recover from two types of failures: peer failure and proxy failure. In the case of peer failure, a peer may suddenly leave the system without notifying the system. In this case, the first affected receiving peer (downstream peer) will declare the unavailability of the lost MDC descriptions using DHT-enabled function, *Remove(key)*. With the use of MDC, most of the peers originally served by the leaving peer

can still continue their video service as long as at least one of the MDC descriptions can still be received from other upstream peers. In the case of proxy failure, if a peer (*i.e.*, index server) issues a request to a failed proxy without any response after a predefined time period, the peer will forward the request directly to the video server.

4. PERFORMANCE EVALUATION

4.1 Simulation Setting

In our simulations, the proposed caching scheme MP³ is compared with PROP, a recently proposed P2P-assisted proxy caching algorithm [11]. All users are assumed to arrive according to a random Poisson distribution. Each user may quit or continue to stay in the system after the requested video service has completed. The quitting probability is set to 50%. The number of videos tested is set to 30. The length of each video file is assumed to be 600 seconds, and each video file is assumed to be encoded into four MDC descriptions with different sizes [10]. The access popularity of all videos follows a Zipf-like distribution [1]. The commonly used skew factor for media-on-demand service is 0.271 [25, 29]. The receiving bandwidth of all peers is assumed to have a uniform distribution. The caching space of peers is assumed to be the same. We summarize the key parameters used in all simulations in Table 2.

Table 2. Parameters used in simulations.

Parameters	Default value
Number of videos	30
Skew parameter of Zipf-like popularity distribution	0.271
Mean user arrival rate λ of Poisson distribution	4
Length of a video (sec.)	600
Number of MDC descriptions	4
Size of a complete segment (KB)	640
Size of a data chunk (<i>i.e.</i> , an MDC description of a complete segment) (KB)	64, 128, 192, 256
Peer's receiving bandwidth in uniform distribution (Kbps)	256 to 10,240
Caching space at proxy (MB)	10% of total video size
Caching space at client (MB)	0 to 50
Peer quitting probability	50%
Simulation time (sec.)	3,000

4.2 Simulation Results

As discussed in section 2.3, PROP does not consider the use of scalable video coding. In our first simulation, we assume that PROP employs non-scalable video coding. To evenly compare PROP and MP³ in this scenario, we assume that the extra coding overhead of MDC over non-scalable video coding is 40%. Fig. 3 plots the average observed PSNR of all peers versus peer receiving bandwidth. Obviously, MP³ is more adaptive since it can provide different qualities for all peers; on the other hand, PROP cannot serve low-bandwidth peers whose receiving bandwidth is below 1Mbps.

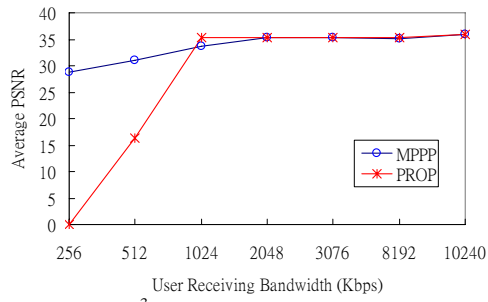


Fig. 3. MP³ vs. PROP: avg. PSNR observed by all peers.

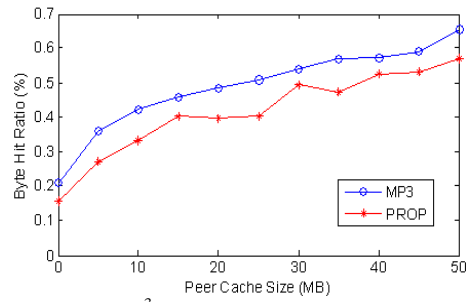


Fig. 4. MP³ vs. PROP: overall byte hit ratio.

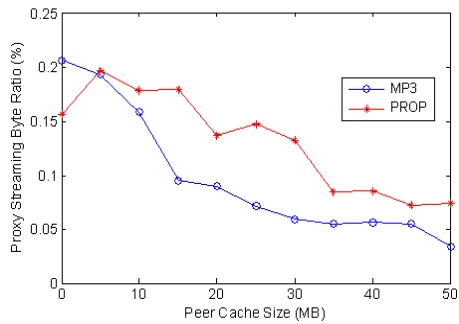


Fig. 5. MP³ vs. PROP: proxy streaming byte hit ratio.

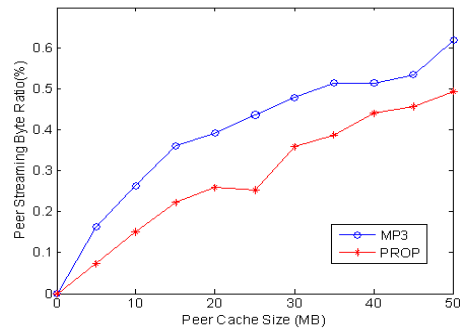


Fig. 6. MP³ vs. PROP: peer streaming byte hit ratio.

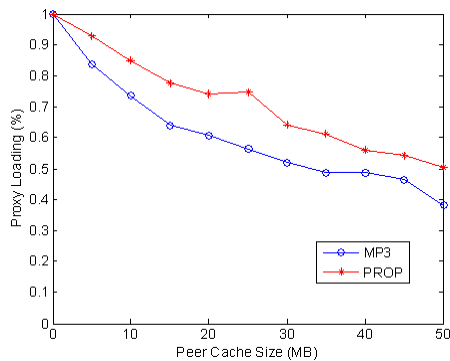


Fig. 7. MP³ vs. PROP: proxy loading.

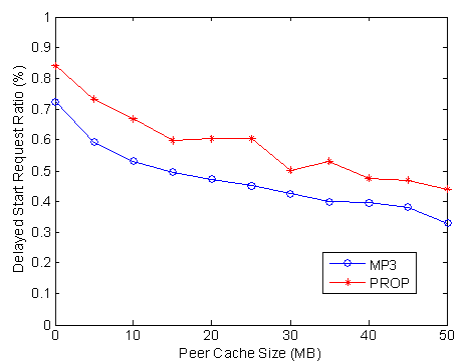


Fig. 8. MP³ vs. PROP: delayed start request ratio.

Next, we assume that PROP can take advantage of MDC by treating the four MDC descriptions as four independent video files. Fig. 4 shows the overall byte hit ratio versus peer cache size for both MP³ and PROP. Overall byte hit ratio is defined as the ratio of the total number of data chunks provided by both the proxy and all peers to the total number of data chunks requested by all clients. It can be observed that byte hit ratio increases with peer cache size, and MP³ outperforms PROP for all cache sizes. For exam-

ple, when the cache size of each peer is set to zero, MP³-Proxy can satisfy 20% of data requests, while PROP-Proxy can satisfy only 16% of data requests. When the cache size of each peer is set to 50MB, MP³ also outperforms PROP by 9% in terms of byte hit ratio.

Fig. 5 shows the proxy streaming byte hit ratio versus peer cache size for both MP³ and PROP. Proxy streaming byte hit ratio is defined as the ratio of the total number of data chunks provided by the proxy to the total number of data chunks requested by all clients. It can be observed that proxy streaming byte hit ratio decreases with peer cache size. This is so since when peer cache size increases, peers can share more load originally imposed on the proxy. It can also be observed that the proxy caching algorithm of MP³ outperforms that of PROP since MP³-proxy has a lighter load than PROP-proxy.

Fig. 6 shows the peer streaming byte hit ratio versus peer cache size for both MP³ and PROP. Peer streaming byte hit ratio is defined as the ratio of the total number of data chunks provided by all peers to the total number of data chunks requested by all peers. It can be observed that peer streaming byte hit ratio increases with peer cache size. It indicates that when peer cache size increases, peer caching algorithm do make peers share more load of the system. This is an attractive effect because the aggregate caching space of all peers is significant, which can largely improve overall system performance. It can also be observed that the peer caching algorithm of MP³ outperforms that of PROP since MP³ peers share more load than PROP peers.

Fig. 7 depicts the normalized proxy loading versus peer cache size for both MP³ and PROP. Proxy loading is defined as the total number of bytes provided by the proxy plus the total number of bytes fetched by the proxy. It is normalized by two times the total number of bytes required by all clients, which is the maximal load that is possible on the proxy. It can be observed that MP³ outperforms PROP in all cases. For example, when peer cache size is set to 50MB, the proxy loading of MP³ is about 38%, while that of PROP is about 50%.

Fig. 8 shows the delayed start request ratio versus peer cache size for both MP³ and PROP. Delayed start request ratio is defined as the ratio of the total number of requests served by the video server to the total number of requests issued by all clients. It can be observed that the delayed start request ratio decreases with the peer cache size. On average, MP³ outperforms PROP by 11%. This is because MP³ takes the relative locations of data chunks in a video file into account, while PROP does not consider this factor.

5. CONCLUSION

The state-of-the-art development of video caching employs a P2P-assisted proxy architecture. Under this architecture, we make use of MDC and develop a new caching scheme, MP³, for both the video proxy and peers. With the use of MDC, MP³ is highly suitable for a heterogeneous network environment. We design MP³ caching algorithms from many points of view, such as users' startup latency, receiving bandwidth, and caching space, video transmission cost, and video quality. Our simulation results show that MP³ can effectively provide adaptive video quality for heterogeneous clients and can significantly reduce the load imposed on both the video proxy and the underlying networks.

REFERENCES

1. L. Breslau, C. Pei, F. Li, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies*, 1999, pp. 126-134.
2. S. Chen, B. Shen, S. Wee, and X. Zhang, "Adaptive and lazy segmentation based proxy caching for streaming media delivery," in *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2003, pp. 22-31.
3. D. J. Deng and H. C. Yen, "Quality-of-service provisioning system for multimedia transmission in IEEE 802.11 wireless LANs," *IEEE Journal on Selected Areas in Communications*, Vol. 23, 2005, pp. 1240-1252.
4. D. J. Deng, L. W. Chang, T. Y. Wu, and C. C. Hu, "Guaranteed QoS provision scheduling mechanism for CBR traffic in IEEE 802.16 BWA systems," *Journal of Internet Technology*, Vol. 9, 2008, pp. 403-409.
5. D. J. Deng, C. H. Ke, Y. M. Huang, and H. H. Chen, "Delay analysis and performance evaluation of a guaranteed QoS provisioning mechanism for real-time traffic in IEEE 802.16e BWA systems," in *Proceedings of International Wireless Communications and Mobile Computing Conference*, 2008, pp. 326-331.
6. J. W. Ding and Y. M. Huang, "Resource-based striping: An efficient striping strategy for video servers using heterogeneous disk-subsystems," *Multimedia Tools and Applications*, Vol. 19, 2003, pp. 29-51.
7. J. W. Ding, S. Y. Tseng, and Y. M. Huang, "Packet permutation: A robust transmission scheme for continuous media streams over the Internet," *Multimedia Tools and Applications*, Vol. 21, 2003, pp. 281-305.
8. J. W. Ding, C. T. Lin, and S. Y. Lan, "A unified approach to heterogeneous video-on-demand broadcasting," *IEEE Transactions on Broadcasting*, Vol. 54, 2008, pp. 14-23.
9. T. T. Do, K. A. Hua, and M. A. Tantaoui, "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Proceedings of IEEE International Conference on Communications*, 2004, pp. 1467-1472.
10. F. H. P. Fitzek, B. Can, R. Prasad, M. Katz, and D. S. Park, "Traffic analysis of multiple description coding of video services over IP networks," in *Proceedings of the 7th International Symposium on Wireless Personal Multimedia Communications*, 2004, pp. 266-270.
11. L. Guo, S. Chen, and X. Zhang, "Design and evaluation of a scalable and reliable P2P assisted proxy for on-demand streaming media delivery," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, 2006, pp. 669-682.
12. Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P²Cast: Peer-to-peer patching scheme for VoD service," in *Proceedings of the 12th International Conference on World Wide Web*, 2003, pp. 301-309.
13. M. Y. Hsieh, Y. M. Huang, and T. C. Chiang, "Transmission of layered video streaming via multi-path on ad hoc networks," *Multimedia Tools and Applications*, Vol. 34, 2007, pp. 155-177.
14. Y. M. Huang, J. W. Ding, and S. L. Tsao, "Constant time permutation: An efficient block allocation strategy for VBR continuous media data," *The VLDB Journal*, Vol.

- 8, 1999, pp. 44-54.
15. Y. M. Huang, M. Y. Hsieh, and M. S. Wang, "Reliable transmission of multimedia streaming using a connection prediction scheme in cluster-based ad hoc networks," *Computer Communications*, Vol. 30, 2007, pp. 440-452.
 16. A. T. S. Ip, J. Liu, and J. C. S. Lui, "COPACC: An architecture of cooperative proxy-client caching system for on-demand media streaming," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, 2007, pp. 70-83.
 17. C. F. Kao and C. N. Lee, "Aggregate profit-based caching replacement algorithms for streaming media transcoding proxy systems," *IEEE Transactions on Multimedia*, Vol. 9, 2007, pp. 221-230.
 18. C. W. Kong and Y. B. Lee, "Slice-and-patch – An algorithm to support VBR video streaming in a multicast-based video-on-demand system," *Journal of Information Science and Engineering*, Vol. 19, 2003, pp. 517-530.
 19. C. T. Lin and J. W. Ding, "CAR: A low latency video-on-demand broadcasting scheme for heterogeneous receivers," *IEEE Transactions on Broadcasting*, Vol. 52, 2006, pp. 336-349.
 20. J. Liu, S. C. Yang, H. F. Yu, and L. M. Tseng, "Content delivery network with hot-video broadcasting and peer-to-peer approach," *Journal of Information Science and Engineering*, Vol. 20, 2004, pp. 1125-1139.
 21. J. Liu, X. Chu, and J. Xu, "Proxy cache management for fine-grained scalable video streaming," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004, pp. 1490-1500.
 22. X. Liu, and S. T. Vuong, "Supporting low-cost video-on-demand in heterogeneous peer-to-peer networks," in *Proceedings of the 7th IEEE International Symposium on Multimedia*, 2005, pp. 523-533.
 23. E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys and Tutorials*, Vol. 7, 2005, pp. 72-93.
 24. P. Seeling, M. Reisslein, and B. Kulapala, "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial," *IEEE Communications Surveys and Tutorials*, Vol. 6, 2004, pp. 58-78.
 25. Z. Xiang, Q. Zhang, W. Zhu, Z. Zhang, and Y. Q. Zhang, "Peer-to-peer based multimedia distribution service," *IEEE Transactions on Multimedia*, Vol. 6, 2004, pp. 343-355.
 26. D. Xu, S. S. Kulkarni, C. Rosenberg, and H. K. Chai, "Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, Vol. 11, 2006, pp. 383-399.
 27. W. P. K. Yiu, J. Xing, and S. H. G. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE Journal on Selected Areas in Communications*, Vol. 25, 2007, pp. 1717-1731.
 28. H. C. Yang, H. F. Yu, and L. M. Tseng, "Adaptive live broadcasting for highly-demand video," *Journal of Information Science and Engineering*, Vol. 19, 2003, pp. 531-549.
 29. Q. Zhang, Z. Xiang, W. Zhu, and L. Gao, "Cost-based cache replacement and server selection for multimedia proxy across wireless Internet," *IEEE Transactions on Multimedia*, Vol. 6, 2004, pp. 587-598.



Jen-Wen Ding (丁建文) received his B.S., M.S., and Ph.D. degrees in Engineering Science from National Cheng Kung University, Tainan, Taiwan, in 1996, 1998, and 2001, respectively. He joined National Kaohsiung University of Applied Sciences from 2004. He is currently an Assistant Professor of Department of Information Management, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan. His research interests include multimedia communications, peer-to-peer computing, and mobile computing.



Shi-Yuan Lan (籃士淵) received his B.B.A. and M.B.A. degrees in Information Management from National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, in 2005 and 2007, respectively. His research interests include computer networks and multimedia communications.