Front. Comput. Sci. China, 2008, 2(3)
DOI 10.1007/s11704-009-0000-0

RESEARCH ARTICLE

PopCap: Popularity oriented Proxy Caching for Peer-assisted Internet On-demand Video Streaming Services
Proxy Caching for Peer-assisted Internet VoD

Ye, TIAN(✉)[1], Bangchuan, LIU[1], Zhenhua, HE[1]

1 Anhui Province Key Laboratory on High Performance Computing and Application,
School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230027, China

Abstract With the success of the Internet on-demand video (VoD) streaming services, the bandwidth required and the cost incurred on the video server become extremely large. Peer-to-peer (P2P) network and proxy are two common ways for reducing the server's workload. In this paper, we consider a peer-assisted Internet VoD system with proxies deployed at the domain gateways. We formally present the video caching problem with the objectives of reducing the video server's workload and avoiding the inter-domain traffic, and obtain its optimal solution. Inspired by the theoretical analysis, we develop a practical protocol named *PopCap* for Internet VoD services. Comparing with previous works, PopCap is does not require additional infrastructure support, works inexpensively and is able to cope well with the workload characteristics of the Internet VoD service. From simulation-based experiments driven by real-world datasets from YouTube [1], we find that PopCap can effectively reduce the video server's workload, therefore provides a superior performance regarding the video server's traffic reduction.

Keywords Internet Video-on-Demand (VoD), Peer-to-Peer(P2P), Caching, Algorithm/protocol design and analysis

1 Introduction

With the help of the Web2.0 techniques, interactive information sharing using audio and video instead of plain text becomes more and more popular on today's Internet. Among the newly emerging Web2.0 applications, the Internet on-demand video (VoD) streaming services, such as YouTube [1], and the Chinese-based Tudou [2] and Youku [3], have attracted many users. Unlike the traditional IP-layer VoD [4] and the recently popular P2P VoD [5], in Internet VoD, videos are contributed by users, and people can upload, view, mark, and comment the videos. Because of its openness and interactivity, Internet VoD rapidly becomes very popular after its birth, for example, it was reported in 2006 that each day there were more than 65,000 new videos uploaded on YouTube, and the site was receiving 100 million video views per day [6]. It is also estimated that in 2007 YouTube consumed the bandwidth of the entire Internet in 2000 [7]. With its extraordinary large video collection and user views, Internet VoD is far from an online cinema. For such a large system, how to efficiently and inexpensively deliver videos to users becomes a very challenging problem.

Currently nearly all the existing Internet VoD systems adopt a client-server architecture, where all the videos are uploaded by the video server (server cluster or CDN) to users, thus the bandwidth required and the cost incurred is very large. For example, in 2008 YouTube was estimated to pay about one million US$ per day for the bandwidth [8]. Obviously, it is economic and technical beneficial if the traffic on the video server could be reduced. Recently, some researchers propose to use a P2P network (e.g. [9]), where peers cache their downloaded videos and help to distribute them, for Internet VoD services. However, due to the special features of the Internet VoD service, technologies that are widely used in

Received month dd.yyyy; accepted month dd.yyyy

E-mail: *yetian@ustc.edu.cn*, {*liuchuan*, *ehzh*}@*mail.ustc.edu.cn*

P2P VoD streaming (e.g. PPLive [5]) may not work well under the context of Internet VoD. On the other hand, in traditional VoD services, proxies have been widely used(e.g., [10] [11]). In this paper, we consider to combine the two, and propose a protocol for the peer-assisted Internet VoD system with proxies.

In this work, we first examine the characteristics of Internet VoD's workload by investigating real-world datasets obtained from YouTube. We find that under Internet VoD, there exists an extreme imbalance regarding the videos' popularity, and all the videos are very short. We then formally present the video caching problem of the system combining proxy and P2P network, with the objectives of reducing the video server's workload and avoiding the inter-domain traffic. We show that for such a problem, an optimal solution exist. With the awareness of the Internet VoD service's workload characteristics and inspired by the theoretical analysis, we design *PopCap*, a practical protocol for the proxy and the peers in the P2P network to independently and collaboratively cache videos. In the PopCap protocol, videos are cached on the proxy in a proactive way, while for the video caching on peers, we use blocking as well as evicting to cope with the extremely imbalanced video popularity, therefore enable the peers to avoid globally excessive or inadequate caching of the videos. Unlike many P2P-based caching systems such as PROP [11], PopCap does not rely on a DHT-based overlay, therefore can cope with the Internet VoD's characteristics well, while unlike traditional Internet VoD systems, PopCap exploits the resource on individual peers, thus extensively reduces the server's workload. By comparing PopCap with existing solutions. we find that PopCap is more practical and inexpensive, which makes it suitable to be deployed under the Internet VoD service. From simulation-based experiments driven by real-world YouTube datasets, we find that PopCap protocol could effectively reduce the video server's workload by making a better use of the caching spaces on the proxy and the peers, moreover, its "smart update" mechanism provides flexibility to further reduce the video server's overall bandwidth cost.

The remainder part of this paper is organized as the follows: Section 2 introduces the related works; Section 3 describes the architecture of the peer-assisted Internet VoD system under discussion; In Section 4, characteristics of the Internet VoD service are analyzed using real-world datasets; In Section 5, we formally present the video caching problem, obtain its optimal solution, and discuss its implications; We propose the PopCap protocol in Section 6; In Section 7 we investigate the performance of PopCap and compare it with other solutions; Finally, we conclude this work in Section 8.

2 Related Work

With its great commercial success and influence on the Internet, there are many works studying Internet VoD in recent years. In [12], YouTube and another popular Internet VoD service in Korea are studied, and the authors analyze many aspects of the service including life-cycle of the videos and its relationship with the video requests. It is also shown that the server's workload could be greatly reduced if some P2P assistance is available. In [9], traces from the MSN video service [13] are investigated, and with a simple analytical model, it is shown that the traffic on the video server could be dramatically reduced if a P2P network helps to distribute the videos, even if a strong locality rule is applied for the P2P network. In [14], by exploring the data obtained from YouTube, social network patterns are observed among the videos, and the authors propose a novel P2P-assisted video delivering framework that explores the clustering of the video social network for improving the playback quality and reducing the server's workload. In [15], the network traffic caused by campus users downloading videos from YouTube is investigated, and the authors point out that by smartly exploiting metadata, better video caching strategies could be developed.

On-demand video streaming using P2P techniques also becomes very popular in recent years. Examples include P2Cast [16], P2VoD [17] and DSL [18]. P2Cast and P2VoD investigate a tree-based overlay structure to organize the peers, and DSL presents a dynamic skip list overlay to enable the VCR operation. Cui et al. [19] propose oStream, which extends application multicast to support VoD with buffers on peers. Tian et al. [20] consider a probabilistic caching mechanism on the clients to reduce the video server's workload.

On the other hand, deploying dedicated proxies for reducing the video server's workload and providing a better quality of streaming service has been studied for a long time. In [10], the conflict between hit ratio and proxy jitter in the proxy caching strategy is investigated, and a new proxy system named Hyper Proxy is proposed. In [21], a cooperative proxy-client caching system is proposed, where low cost of P2P network and robustness of dedicated proxy are combined. In a recent work [22], the caching problem for peers in a P2P assisted VoD system is investigated, and an algorithm with the feature of proportional partial admission and eviction of video segments is proposed. In [11], the authors consider a VoD service combining both proxy and P2P network, and propose a system named PROP to reliably and scalably cache and distribute the videos.

Our work is different from previous works in that we jointly consider the proxy and the peer caching under the context of the Internet VoD service. By combining proxy with P2P network, critical issues such as peer-proxy collaboration must be addressed; while by considering the problem under the context of Internet VoD, characteristics of this application must be taken into consideration and technologies that are widely applied in ordinary P2P VoD systems must be carefully re-considered and re-examined before get applied under Internet VoD.

3 The System Architecture

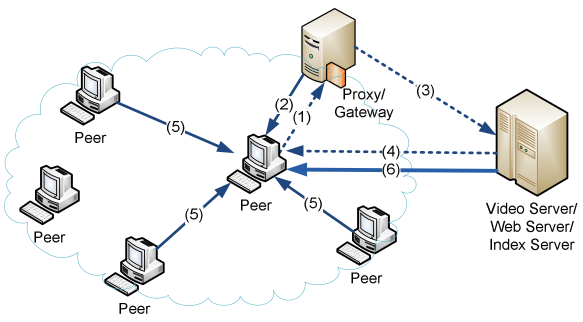


Fig. 1 Demonstration of system architecture for peer-assisted Internet VoD with proxy caching

In this paper, we consider a peer-assisted Internet VoD system with proxies. Typically in such a system there are three components: (a) the server, which contains at least a web server, an index server, and a video server (server cluster or CDN); (b) the end-system clients which request and download the videos; and (c) the proxy which is deployed at the gateway of a domain and uploads the cached video to the clients residing in the same domain. In addition, clients form a self-organized P2P overlay network, in which each of them is a peer and independently maintains a video cache. Generally, the server is maintained by the video service provider (VSP) such as YouTube [1], the proxy can be runned by VSP or ISP, and the peers are autonomous ordinary end-systems. A demonstration of the system architecture is shown in Fig. 1.

In a peer-assisted Internet VoD system, a peer caches the videos it has downloaded. Peers independently manage their locally cached videos. When a peer joins or leaves the system, or a video replica is cached or gets evicted, the peer reports to the index server. As demonstrated in Fig. 1, when a peer requests a video, for example, by clicking the video's URL on the VSP's website, if a proxy is available for the domain of the peer, the gateway redirects the request to the proxy (step 1), and the proxy uploads to the peer if it has cached a replica and has enough outgoing bandwidth (step 2). If there is no proxy or the proxy is unable to upload the video, the request is then sent to the index server (step 3), which returns with a list containing some other peers on the P2P network that currently have this video cached (step 4). The peer then requests and downloads the video from some of these peers in a P2P manner (e.g. swarming [23]) (step 5). Finally, in case that there is no peer that have cached this video, or none of the index server returned peers can upload the video due to the reasons such as poor network condition or stale information on the index server, the requesting peer directly downloads the video from the video server (step 6). From the procedure, we can see that by deploying a proxy, some inter-domain traffic can be avoided, as both P2P sharing and direct downloading from the video server incur traffics out of the domain. Moreover, the proxy and the P2P network can reduce the workload on the video server. Clearly, to effectively achieve these objectives, proxy and the P2P network should cache the videos in a smart and cooperative way. In the following sections, we will investigate this problem from theoretical as well as practical protocol designing aspects.

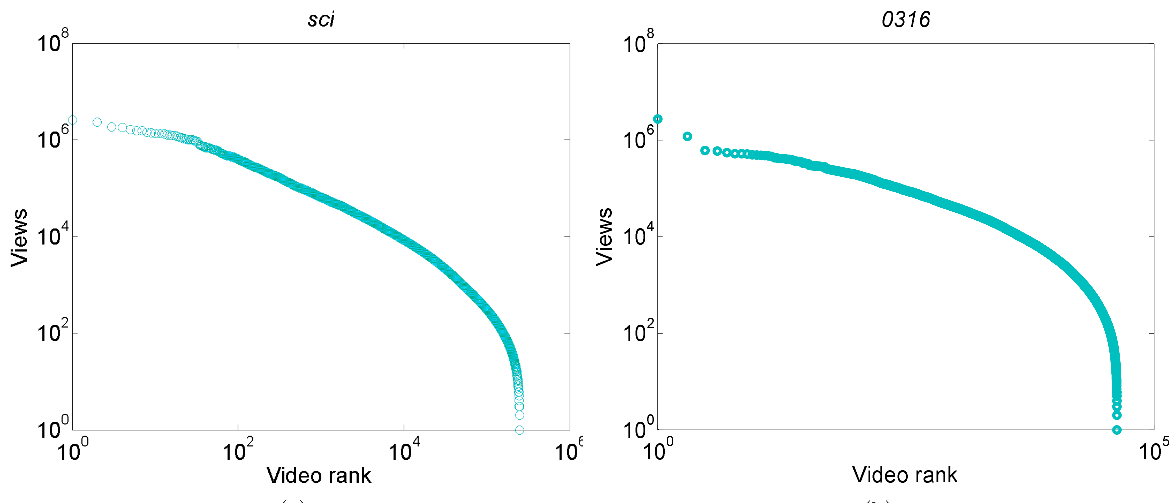
4 Characterizing Internet VoD

Before analyzing the video caching problem, we first investigate some characteristics of the Internet VoD service. Two datasets collected by crawling YouTube, i.e. the one used in [12] and the one used in [14]), are used for our investigation. For first one, we uses the "sci" dataset which contains 252,255 videos. For the data from [14], we choose the dataset collected on Mar. 16th, 2007 containing 42,628 videos (referred to as the "0316" dataset).

We first examine the popularity of the videos in Internet VoD. In Fig. 2, we plot the view times against their ranks for all the videos in the "sci" and the "0316" datasets . Note that in the figure, the curves are plotted under log-log scale. In recent years, many works consider the video access pattern to follow a Zipf distribution (e.g. [11] and [24]) or some other Zipf-like distributions, such Zipf with exponential cutoff [12] and Mandelbrot-Zipf [22]; on the other hand, a recent work reveals that popularity of the videos is more likely to follow a stretched exponential distribution [25]. In this work, we do not try to fit the video access data from the datasets with theoretical models, but focus on its fundamental feature. That is, we find from the figure that extreme imbalance exits regarding videos' popularity. For example, in the "sci" dataset, the most popular video gets viewed 2,537,904 times, while the mean and the median view times in this dataset are 2,140 and 186 respectively; for the "0316" dataset, the values are 2,755,993, 5,405, and 838 respectively.

Another feature of the Internet VoD service we are interested is the video length. In [14], it is reported that majority of the videos on YouTube are no longer than 700 seconds, while by examining the datasets, we find that the mean video lengths are 143 and 205 seconds for "sci" and "0316" respectively.

In summary, by examining the datasets obtained from



(a) (b)

Fig. 2 Popularity for the videos in the (a) "sci" dataset and the (b) "0316" dataset

YouTube, we find that: 1) there exists *an extreme imbalance* regarding the video popularity; 2) comparing with traditional VoD, videos in Internet VoD are *very short*. The two features of the Internet VoD service impose a great challenge for applying P2P network-based technologies on Internet VoD, for the following reasons: First of all, with very short videos, peers will perform operations such as video request, cache update and cache announcement very frequently, this incurs a great workload on the P2P overlay; Second, with the extremely imbalanced video popularity, the workload on a DHT-based overlay very also be extremely imbalanced. For example, the peer that manages the key of the most popularly video will have a much larger workload comparing with other peers.

5 Analyzing Video Caching in Internet VoD

In this section, we formally present the video caching problem for a peer-assisted Internet VoD system and theoretically analyze it.

5.1 The Video caching problem

In our theoretical analysis, we consider a peer-assisted Internet VoD system with a collection of M videos, which are ranked in a descending order on their popularity, and there are N online peers. For simplicity, we assume that all the videos are equal-sized. For each peer, it can cache up to c video replicas, while for the proxy, it can cache $C(C >> c)$ videos. For each video, say video i, it may be cached by the proxy, and it could also be cached by a number of the online peers. We use $c_i(c_i = 0, 1)$ to denote whether or not video i is cached by the proxy, that is, if cached, $c_i = 1$, and $c_i = 0$ if not cached. We also use $n_i(N \geq n_i \geq 0)$ to denote the number of the peers that currently caches the video. For all the M videos, a vector $\vec{n} = \{n_1, n_2, ..., n_M\}$ is used to denote the caching status of the P2P network and a vector $\vec{c} = \{c_1, c_2, ..., c_M\}$ is used to denote the proxy's caching decision.

In our analysis, we assume that the proxy has sufficient out-going bandwidth and never fails, thus when a video is cached by the proxy, the proxy can upload to any requesting peer. On the other hand, peers in the P2P network are different. A peer may fail, or it may evict the video to make room for a new video replica, but the index server that keeps the video's caching status may have stale information. Moreover, even for a peer that does have the video cached, the network condition may be very poor between the peer and the requesting peer. To accommodate these concerns, in our analysis we simply use a probability of p to denote the chance that a peer which is supposed to be able to serve a video by the index server actually is unable to upload the video. Obviously with these unreliable peers, the probability that video i could not be served by the P2P network is $(1-p)^{n_i}$.

Let λ be the total video request rate from the N online peers, then the rate of the requests that goes to the video server could be expressed as $\sum_{i=1}^{M} \lambda p_i(1-c_i)(1-p)^{n_i}$. If we define $\rho(\vec{n}, \vec{c}) = \sum_{i=1}^{M} p_i(1-c_i)(1-p)^{n_i}$ as the ratio of the workload on the video server, then the video caching problem for the peer-assisted Internet VoD with proxy can be expressed as

$$\begin{array}{ll} Minimize & \rho(\vec{n},\vec{c}) = \sum_{i=1}^{M} p_i(1-c_i)(1-p)^{n_i} \\ s.t. & c_i = 0,1; i = 1,2,...,M \\ & n_i \geq 0; i = 1,2,...,M \\ & \sum_{i=1}^{M} c_i = C \\ & \sum_{i=1}^{M} n_i = N \times c \end{array} \quad (1)$$

5.2 The optimal solution and its implications

We first consider the case that no proxy is deployed. For this case, the problem in Equation (1) can be rephrased as

$$\begin{array}{ll} Minimize & \rho(\vec{n}) = \sum_{i=1}^{M} p_i(1-p)^{n_i} \\ s.t. & \sum_{i=1}^{M} n_i = N \times c \end{array} \quad (2)$$

For this problem, we have the following result.

Theorem 1. For the video caching problem in Equation (2), the optimal solution $\vec{n}^*$ is

$$n_i^* = \frac{N}{M}c - \frac{\log p_i}{\log(1-p)} + \frac{\log \prod_{j=1}^{M} p_j}{M \log(1-p)} \quad (3)$$

Proof. First of all, for n_i^* in Equation (3), it is easy to see that $\sum_{i=1}^{M} n_i^* = Nc$, suggesting that Equation (3) is a feasible solution. Furthermore, for any i and j, $i \neq j$, we have

$$p_i(1-p)^{n_i^*} = p_j(1-p)^{n_j^*}$$

We then prove that the solution in Equation (3) is local optimal. To show this, we consider another solution $\vec{n}' = \{n_1', n_2', ..., n_M'\}$, where for particular i and j ($i \neq j$), $n_i' = n_i^* + 1$ and $n_j' = n_j^* - 1$, and for any k ($k \neq i, j$), $n_k' = n_k^*$. By taking $\vec{n}'$ and $\vec{n}^*$ into the objective function $\rho(\vec{n})$ in Equation (2) respectively, we have

$$\begin{aligned} &\rho(\vec{n}') - \rho(\vec{n}^*) \\ &= p_i(1-p)^{n_i'} + p_j(1-p)^{n_j'} - p_i(1-p)^{n_i^*} - p_j(1-p)^{n_j^*} \\ &= p_i(1-p)^{n_i^*}((1-p)-1) + p_j(1-p)^{n_j^*}\left(\frac{1}{1-p} - 1\right) \end{aligned}$$

since $p_i(1-p)^{n_i^*} = p_j(1-p)^{n_j^*}$,

$$\begin{aligned} &\rho(\vec{n}') - \rho(\vec{n}^*) \\ &= p_i(1-p)^{n_i^*}\left((1-p) + \frac{1}{1-p} - 2\right) > 0 \end{aligned}$$

for any $p > 0$. In other words, $\vec{n}^*$ is local optimal.

Finally, note that the objective function $\rho(\vec{n})$ is convex and the constraint is affine, thus the problem in Equation (2) is actually a convex optimization problem. It is well-known that for such a problem, any local optima is global optimal [26], therefore Equation (3) is the optimal solution for the video caching problem without proxy in Equation (2). □

We then consider the video caching problem in Equation (1) when a proxy is deployed, for this problem, we have the following result.

Theorem 2. For the video caching problem in Equation (1), the optimal solution is

$$\begin{cases} \begin{cases} n_i^* = 0 \\ c_i^* = 1 \end{cases} ,1 \leq i \leq C \\ \begin{cases} n_i^* = \frac{Nc}{M-C} - \frac{\sum_{j=C+1}^{M}(\log p_i - \log p_j)}{(M-C)\log(1-p)} \\ c_i^* = 0 \end{cases} ,C+1 \leq i \leq M \end{cases} \quad (4)$$

Proof. We first show that for any solution of $\vec{c}$, i.e., the proxy selects any C videos from the the M videos to cache, the optimal solution of $\vec{n}$ is

$$\begin{cases} n_i^* = 0, & \text{if } c_i = 1 \\ n_i^* = \frac{Nc}{M-C} - \frac{\log p_i}{\log(1-p)} + \frac{\log \prod_{j,c_j=0} p_j}{(M-C)\log(1-p)}, & \text{if } c_i = 0 \end{cases}$$

To show this, note that for any video, say video i, when it is cached by the proxy, $c_i = 1$, $\lambda p_i(1-c_i)(1-p)^{n_i} = 0$ regardless of the value of n_i. So for this video, proxy will serve all the requests, and it is not necessary for the P2P network to cache any replicas, i.e., $n_i^* = 0$. Now with C videos being cached by the proxy, $M-C$ videos with $c_i = 0$ are for the P2P network to cache. According to Theorem 1, for these videos, the optimal solution is

$$n_i^* = \frac{Nc}{M-C} - \frac{\log p_i}{\log(1-p)} + \frac{\log \prod_{j,c_j=0} p_j}{(M-C)\log(1-p)}$$

We next show that with $\vec{n}^*$, the optimal solution for $\vec{c}$ is

$$\begin{cases} c_i = 1, 1 \leq i \leq C \\ c_i = 0, C+1 \leq i \leq M \end{cases}$$

We prove this result as the following. By taking $\vec{n}^*$ into the objective function $\rho(\vec{n},\vec{c})$, we can see that

$$\begin{aligned} \rho(\vec{n}^*,\vec{c}) &= \sum_{i,c_i=0} p_i(1-p)^{n_i^*} \\ &= (M-C)(1-p)^{\frac{Nc}{M-C}}\left(\prod_{i,c_i=0} p_i\right)^{\frac{1}{M-C}} \end{aligned}$$

To minimize $\rho(\vec{n}^*,\vec{c})$, we just need to minimize $\prod_{i,c_i=0} p_i$. Obviously, the best way is to set $c_i = 0$ for the $M-C$ least popular videos. In other others, the proxy caches the C most popular videos. Therefore we have proved the theorem. □

In our problem formulation in Equation (1), we only consider the objective of minimizing the video server's workload (i.e., minimizing ρ), but do not consider the objective of avoiding the inter-domain traffic. However, we can see that the solution obtained in Equation (4) also achieves this objective. As both the direct downloading from the video server and using the P2P network incurs inter-domain traffic, clearly the most requested videos should be cached by the proxy to avoid the inter-domain traffic to the greatest extent, as shown in Equation (4).

In a practical peer-assisted Internet VoD system, clearly it is infeasible obtain the values of the parameters pi, p, N, therefore we can not apply the solution in Equation (4) directly. However, some insights could be obtained from Equation (4). First of all, note that in the optimal solution, proxy only caches the C most popular videos, while peers do not cache any replica of them. This observation suggests that in a practical system, proxy must be able to identify these popular videos while peers must be able to avoid caching these videos. Second, we note that for the videos that are not cached by proxy, a proper number of replicas should be cached by peers in the P2P network, in particular, the difference of the replica numbers for video i and video j is proportional to $\log p_i - \log p_j$. This observation suggests that if we know how to cache one video, say video k, then for any specific video, in principle we will know how to cache it by using video k as a benchmark.

5.3 Numerical evaluation

Finally, we numerically evaluate the effectiveness of our solution for peer-assisted Internet VoD. For video popularity, we use the data of the first 20,000 videos obtained from the YouTube "sci" dataset as shown in Fig. 2. We calculate the minimized workload ratios on the video server by applying the optimal solution on the objective function in Equation (2). For other parameters, we let $N = 2,000$, $p = 0.8$, $C = 1,000$, and $c = 5$ as the default.

We first investigate the influence of the proxy cache size. Under varying proxy cache size C we plot the video server ratio ρ against C in Fig. 3(a). From the figure one can see that by enlarging the cache size, more videos could be served by the proxy. Moreover, we can see that the curve is nearly linear, which means if the cost for larger storage does not increase as much as the cost for network traffic, it is economic beneficial for ISP or VSP to pay for the proxy storage than for the bandwidth.

We also consider the influence of the peer cache size by varying c and plot the video server ratio ρ against c in Fig. 3(b). From the figure we can see that when c is small and gets increased, ρ decreases dramatically, and when c is large, ρ approaches zero. This observation indicates that P2P networking is promising for the Internet VoD service, thus it is very important to encourage the peers, which are usually selfish, to contribute their local storage in the P2P network.

We calculate the server's workload ratio ρ under varying peer reliability values of p and plot ρ against p in Fig. 3(c). From the figure we can see that even under the moderate peer cache size, increasing p can dramatically decrease ρ and reduce the server's workload, therefore it is essential to timely update the index server and eliminate the stale information.

6 The PopCap Protocol

Motivated by the theoretical analysis, in this section, we consider under the real-world Internet VoD environment, how to design a practical protocol for the proxy and the peers in the P2P network to independently and cooperatively cache the videos.

6.1 Feasibility of P2P technologies

As shown in Section 4, we have observed two features of Internet VoD by examining real-world datasets: 1) extremely imbalanced video popularity; 2) very short video length. The former incurs a load balancing issue on P2P networks, especially the DHT-based overlays, while the latter causes a great increase of workload. Because of these observations, before designing the protocol, it is very necessary for us to examine the feasibility of the P2P technologies that are successfully applied in ordinary P2P VoD systems under the context of Internet VoD. It is noted that nearly all the existing P2P VoD systems rely on two essential technologies: the swarming overlay technology and the DHT-based overlay technology, where the former is used to distributed the video data and the latter is applied for resource lookup and management.

In previous efforts of applying P2P networks on the Internet VoD service, P2P swarming technology has been proved to be able to work well. For example, NetTube [14] applies the swarming protocol similar to the one used in CoolStreaming [27] to enable a peer-assisted Internet VoD streaming. Meanwhile, there are very little efforts of applying DHT-based overlays upon Internet VoD. On the other hand, recently a P2P assisted VoD system named PROP is proposed [11], where a DHT-based overlay is used for assisting peers and the proxy to make the video caching decisions. Therefore, it is very necessary to examine the applicability of the DHT-based overlay technology under the context of Internet VoD.

We consider a peer-assisted Internet VoD system similarly to PROP [11], equipped with a DHT-based overlay. In such a system, when a user finished viewing a video, which has a typical length of 3 minutes, then it may keep the video in its local cache and evict some cached videos to make room, in this case, it must look up the peers that manage the keys of the new cached video and the evicted cached videos on the DHT overlay to make the updates.

Originally, PROP is not proposed for Internet VoD, but for ordinary VoD services. Here we discuss a hypothetic Internet VoD system that applies PROP.

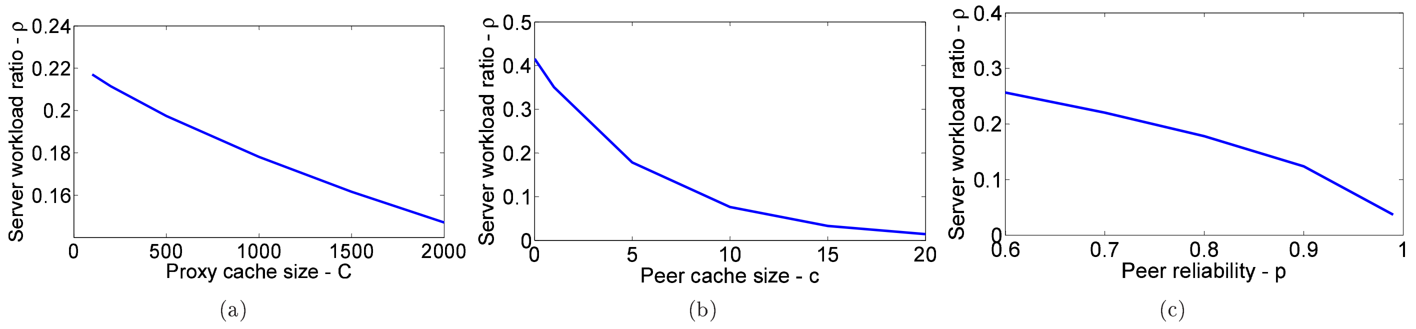


Fig. 3 Video server's workload ratio ρ under optimal proxy and peer caching with (a) varying proxy cache size C, (b) varying peer cache size c, and (c) varying peer reliability p

After that, if the peer requests another video to view, it also must look up the peer that manages the video's key on the DHT overlay to locate the peers that have this video cached. Note that for each DHT lookup, $O(\log N)$ message deliveries are introduced on the overlay, so on average a peer will send or forward $3 \times \log_2 N$ DHT lookup messages every 3 minutes. Suppose that each lookup message is 20 bytes, then for a moderate system containing 5000 peers, on average the bandwidth used for DHT lookup on a peer is $33bps$, which is no longer negligible. Moreover, recall that the popularity of the videos in Internet VoD is extremely imbalanced, which causes an extreme imbalance of the workload on the DHT-based overlay. Suppose that the workload scales with the same rule as the popularity, then by applying the data in "sci", the bandwidth used for DHT lookup on the peer with the heaviest workload will be 39Kbps, which is totally unacceptable. In addition, unlike the transportation of video data, DHT lookups have much more stringent requirement on the delay, while the heavy and imbalanced DHT lookup workload on peers further reduces the efficiency of the DHT-based overlay.

In summary, we find that under the context of the Internet VoD service, DHT-based overlay is not feasible, this makes the systems that rely on DHT (e.g., PROP) no longer suitable for Internet VoD, and forces us to seek approaches which could cope with the features of Internet VoD to practically solve the video caching problem. Following we present *PopCap*, a practical protocol for the proxy and the peers in the P2P network to independently and cooperatively cache videos under a Internet VoD service.

6.2 Metadata collecting and estimation

In PopCap, we use the video server, the proxy, and the peers to collect metadata, the metadata collected will be used for assisting the proxy and individual peer to make their caching decisions. Specifically, for each video, say video i, the video server measures the following metrics:

- n_i^r: total number of the times video i has been requested, (e.g., through a click on the VSP's website);
- t_i^r: the last time video i was requested;
- n_i^s: total number of the times that video i is uploaded by the video server;
- t_i^s: the last time that video i was uploaded by the video server;
- a_i: the time that video i was added to Internet VoD service;
- s_i: video i's size in bytes.

While on the proxy, following metrics are collected:

- n_i^p: total number of the times that video i is uploaded by the proxy;
- t_i^p: the last time that video i was uploaded by the proxy;
- d_i: total bytes of video i that are uploaded by the proxy.

Each peer, say peer j, also keeps the following information for each video replica it currently keeps in its local cache:

- t_i^a: the time that this video replica is added into the peer's local cache.

Note that under current Internet VoD system architecture, the video server, the proxy and the peers only need to use a few counters and time stamps to obtain these metadata. It is not necessary to built a DHT overlay to collect these information.

In addition to the metrics measured directly in the Internet VoD system, two other metrics are estimated. Specifically, for each video, we use the method proposed in [11] to estimate its popularity, that is, for video i, its *popularity* is estimated as

$$P_i = \min\left\{\frac{n_i^r}{t_i^r - a_i}, \frac{1}{t - t_i^r}\right\} \quad (5)$$

where t is the current time. In the expression, $\frac{n_i^r}{t_i^r - a_i}$ is the long-term request rate of this video since the video was added, $t - t_i^r$ is the time since the last request, and

$\frac{1}{t-t_i^r}$ is an approximation of the video's recent request rate.

The proxy also calculates the *usefulness of proxy caching* for each video as the following

$$U_i^{Proxy} = \max\left\{\frac{n_i^p}{n_i^r}, \frac{t-t_i^r}{t-t_i^p}\right\} \tag{6}$$

Here $\frac{n_i^p}{n_i^r}$ is the long-term ratio of the video uploaded by the proxy, and $\frac{t-t_i^r}{t-t_i^p}$ is the likeliness that this video will be uploaded by the proxy on the next request.

6.3 Proxy cache strategy

The optimal solution in Equation (4) shows that for a peer-assisted Internet VoD system, proxy should cache the most popular videos. However, in our analysis it is assumed that all the videos are equal-sized. Clearly this assumption is impractical. For equally popular videos, clearly smaller one is preferred to be cached as caching space is limited. While to reduce the server's workload, for equally popular videos, caching a large one is preferred. Moreover, in Internet VoD, it is possible that a user does not download the entire video, but only download a part of it. Obviously, the videos that are downloaded with a larger portion should also be preferred. In the PopCap protocol, we will consider the three factors in making the caching decisions, that is: 1) the video's popularity; 2) the video's size; and 3) portion of the video that is actually downloaded, in making the caching decisions.

For updating video caches on the proxy, periodically the proxy calculates P_i and U_i for all the videos and solves the following problem

$$\begin{array}{ll} Minimize & \sum_{i=1}^{M} P_i(1-c_i)f(\frac{1}{s_i})g(s_i)h(\frac{d_i}{n_i^p \times s_i}) \\ s.t. & c_i = 0,1; i = 1,2,...,M \\ & \sum_{i=1}^{M} s_i c_i \le C \end{array} \tag{7}$$

In the problem, M is the total number of the videos under consideration, C is the proxy's cache size. $c_i (c_i = 0, 1)$ is the label on whether or not video i should be cached by the proxy. For $f(\frac{1}{s_i})$, $g(s_i)$, and $h(\frac{d_i}{n_i^p \times s_i})$, they represent the proxy's favors to cache small and popular videos for saving the cache space, to cache large and popular videos for reducing the server's workload, and to cache the videos that are more likely to be actually downloaded respectively. For simplicity, in PopCap we let $f(\frac{1}{s_i}) = \frac{1}{s_i}$, $g(s_i) = s_i$, and $h(\frac{d_i}{n_i^p \times s_i}) = \frac{d_i}{n_i^p \times s_i}$. Clearly for the problem in Equation (7), the videos with the largest weights for $(1 - c_i)$ in the objective function should be cached. Therefore, in PopCap's proxy cache strategy, for each video the proxy calculates a weight as

$$W_i = P_i \frac{d_i}{n_i^p \times s_i} \tag{8}$$

and selects the videos with the largest weights to cache, until the cache is full. In PopCap, the proxy periodically determines which videos should be cached, then it requests the missing videos from the video server and discards the videos that should not be cached any longer.

6.4 Peer cache strategy

For PopCap's peer cache strategy, usually there are two approaches for a peer to set up an order in caching videos: *blocking* and *eviction*. For blocking, when a video is downloaded to play, the peer does not necessary to put the video into its local cache, but only caches it with a certain probability; while in eviction, every video is associated with a priority, when a new video needs to be cached, the peer evicts its cached videos based on their priorities. The benefit of blocking is that peers need not to cache the unpreferred videos, but it takes a longer time for a peer to update its cache as not all the chances are exploited. For eviction, the benefit is that all the downloaded videos are cached, however, for very popular videos that have been downloaded very frequently, it is very difficult to reduce the numbers of their replicas even low priorities are assigned. In our peer caching algorithm, we combine the two approaches: we block the popular videos that are likely to be cached by the proxy with a high blocking probability, while the cached videos are evicted according to their priorities. Specifically, for each video, say video i, we use the usefulness of the proxy as its blocking probability, i.e.,

$$Pb_i = U_i^{Proxy} \tag{9}$$

In PopCap, the number of the replicas cached by peers should follow the optimal distribution as shown in Equation (4).To achieve this objective, we carefully control the lifetime of peer cached video replicas. Specifically, for the evicting priority, periodically the proxy calculate a "time to cache" (TTC_i) for each video as

$$TTC_i = \frac{\log P_i - \log P_{min}}{P_i} \tag{10}$$

where P_i is video i's estimated popularity as in Equation (5), and P_{min} is the minimum popularity for all the videos under consideration. When a peer needs to cache a new video, it calculate the priorities for all the videos it has cached as

$$Pr_i = TTC_i - (t - t_i^a) \tag{11}$$

where t is the current time.

The "time to cache" metric indicates how long a video replica should be cached by the P2P network. As from Theorem 2 we known that the number of the video replicas on their popularity is in logarithm, then if the number of the cached replicas for the least popular video is zero, there should be $(\log P_i - \log P_{\min})$ replicas for video i.

Since P_i is also the request rate for the video, according to Little's law [28], the time that a replica is cached is $\frac{\log P_i - \log P_{min}}{P_i}$, which is the replica's "time to cache".

The PopCap's peer cache strategy works as follows: when a peer has downloaded a video, it queries the proxy for the video's blocking probability Pb_i, and caches the video with a probability of $(1 - Pb_i)$. When there is no room for the new video, the peer queries the proxy for TTCs of all its cached videos, and calculates their priorities. Then the peer chooses the one with the smallest priority and evicts, until the newly downloaded video can be cached.

Finally, we compare PopCap's peer cache update strategy with the one used in PROP [11], where a eviction-based mechanism proposed. In PROP, an utility function is calculated by peer on each video, and for very popular videos and for unpopular videos, their utility function values are small while the values for the videos of moderate popularity are relatively large. Peers use the videos' utility function values as the priorities during eviction. However, in PROP, information of exact number of video replicas cached by the P2P network is required, which is collected by a DHT-based overlay. In PopCap, such information is not available as the protocol does not rely on a DHT overlay. In addition, the continuous-valued utility function can not effectively prevent peers to cache the videos that are already being cached by the proxy. For example, suppose that proxy can cache up to C videos, then for the Cth video and the $C + 1$ video regarding the popularity, the utility function will assign values without large difference, therefore peers can not differentiate them while making their caching decisions. But according to our optimal solution in Equation (4), peer's caching decisions on the two videos should be very different. On the other hand, by blocking video with the blocking probability Pb_i, which is not necessarily continuous on the video index, peers can directly use the proxy's caching decisions to make their own decisions. In our experimental study in Section 7, we will see that PopCap can better prevent peers to cache very popular videos than PROP.

6.5 Smart update mechanism

Finally, the timing of the protocol execution is arranged as the following: after every interval of T (T could be a period of time long enough, for example, a week or a month), the proxy calculates P_i, U_i^{Proxy}, and TTC_i for each video. The proxy updates the values of Pb_i and TTC_i for each video at the times of $T, 3T, 5T, ...$, and the proxy calculates P_i and W_i and updates its cache at the times of $2T, 4T, 6T, ...$ In this way, the proxy and the peers update their caches asynchronously: the proxy updates its cache after the peers apply new blocking probabilities and priorities for an interval of T, while the new blocking probabilities and priorities are calculated after the proxy has updated its cached videos and has run for a time of T. In other words, the proxy and the peers let each other to have time to learn and update their caches based on each other's least recent caching decisions. Furthermore, to reduce the traffic on the video server caused by the proxy's cache updating, it is not necessary for the proxy to update at every scheduled time, but the proxy can skip some of them. Specifically, after each update the proxy calculates a significance of the changes as

$$SIG = T \times \sum_{i=1}^{M} (s_i \times W_i \times b_i)$$

where

$$b_i = \begin{cases} 1, c_i(now) - c_i(prev) = 1 \\ 0, otherwise \end{cases}$$

Here b_i the label on whether or not video i is newly cached by the proxy, and SIG is an estimation of the traffic saving on the video server by caching these newly cached videos.

The proxy compare the SIG with the total traffic uploaded by the video server during the recent interval of T as $TRAF$. Specifically, given a threshold, if $SIG < threshold \times TRAF$, the proxy doubles the update interval (e.g., from $2T$ to $4T$, or from $4T$ to $8T$, ..., etc); and if $SIG > threshold \times TRAF$, the proxy halves the interval, until the interval becomes $2T$. We refer to this mechanism as "smart update" of the proxy in the PopCap protocol.

7 Performance Evaluation

In this section, we examine the effectiveness of the proposed PopCap protocol and compare it with existing solutions. An event-driven simulator is developed using C++ for this purpose, and we use the YouTube "sci" dataset [12] and the YouTube "0316" dataset [14] as the video collection of the simulated Internet VoD system in our experiments. But for the "sci" dataset, only the most popular $20,000$ videos are used. In our simulation, time is divided into rounds. During a round, peers request videos according to their popularity, and download them from the proxy, the P2P network or the video server according to the Internet VoD protocol. For simplicity, we use the video length as the size of the video. For the videos on YouTube, the average video length is 185 seconds and we let the default proxy cache size as $2,000$ times of the average video length, and set the default peer cache size as 5 times of the average video length. We also set the default total number of the online peers as $2,000$.

We compare PopCap with PROP [11], which is a protocol for P2P-assisted proxy for large scaled VoD services. In PROP, proxy caches the most popular videos

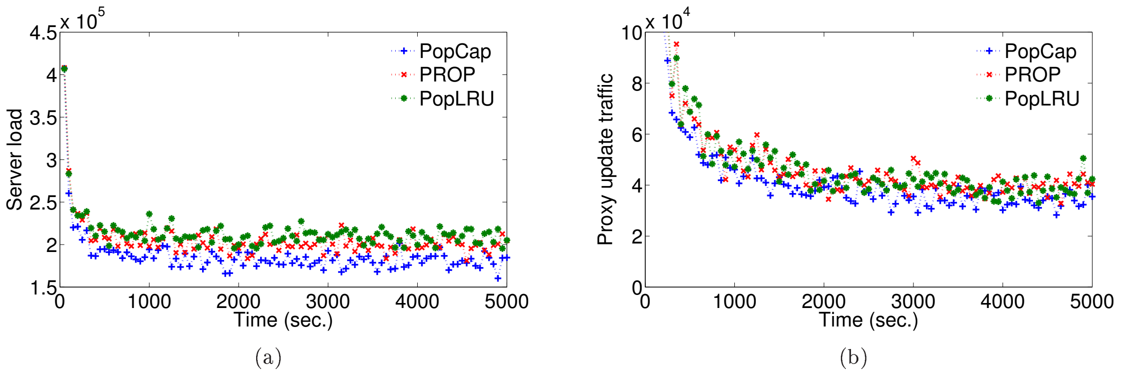


Fig. 4 Evolvement of (a)video server's uploading traffic and (b) proxy's downloading traffic

according to the estimated popularity, and clients update their cached video replicas based on a popularity oriented utility. In PROP, a DHT network is organized by the clients, and for each video, there is a corresponding peer on the DHT network that keeps the information on clients that currently caches this video, as well as the number of replicas currently cached by the P2P network. Although using the DHT technique to find a list of peers that is "good enough" for locating a specific resource is commonly used in many P2P VoD systems, providing accurate number of the peers that currently cache the video is not easy, especially for Internet VoD. For example, when a client has finished playing a video, there will be an event of video caching and a number of evicting events, and the peer must report these events to the corresponding peers on the DHT network. As the average video length on YouTube is only about 3 minutes and usually there are a large amount of online users, there should be a large amount of reports frequently issued by the users. Moreover, as the routing hops for a single report is $O(\log N)$, when N is large, the delay of the peer reports is non-trivial. In other words, for Internet VoD using the DHT technique to obtain the accurate numbers of video replicas is impractical, and expensive. But in PopCap, we do not require the information of the accurate number of the video replicas cached by the P2P network, but only rely on information that could be easily collected as discussed in Section 5. In our simulation, we assume that accurate video replica number is available for PROP.

7.1 Reduction on server's workload

In our first experiment, we compare PopCap with PROP, and another protocol where the proxy caches the popular videos and peers update their caches using the LRU strategy. We start with a proxy randomly caches a number of videos, and the system evolves as the proxy and the peers update their caches. Periodically, we measure the traffic on the video server in uploading the videos to the clients in the cases that the proxy and the P2P network fail to upload, and we also measure the traffic caused by the proxy in updating its cache as well. Fig. 4(a) and (b) shows the how the two traffics evolve with the time under different protocols respectively. We can see from the Fig. 4(a) that among the three protocols, the traffic on the video server under PopCap is the smallest, while PROP outperforms the protocol using the LRU strategy, which conforms to [11]. From Fig. 4(b), PopCap also has the smallest traffic regarding the proxy's updating, but the difference is not significant. From Fig. 4 one can see that both PopCap and PROP has a better performance than the naive protocol of "popularity + LRU", in the remainder part of this section, we only focus on PopCap and PROP.

To have an insight, we investigate how videos are cached by the proxy and the peers under different protocol. During the simulation, at each time of proxy updating, for each video we record: 1) whether or not it is cached by the proxy, if it is cached, we record "1" for this video, otherwise "0"; and 2) by how many peers this video is cached. We refer to the record at a proxy updating time as a "snapshot". We record 50 consecutive snapshots, and by averaging these snapshots we can obtain the video's caching status. Fig. 5 and Fig. 6 show the caching status of the 20,000 videos under PopCap and PROP. For proxy caching, from Fig. 5 we can see that both PopCap and PROP can identify and cache the popular videos, however, in PopCap, we consider popularity as well as the likeliness of actual downloading. In Fig. 6, one can see that peers under PopCap is able to avoid caching very popular videos, as these videos are more likely to be cached by the proxy, while under PROP, although low utilities are assigned to very popular videos, peers still cache them, as these videos are requested very frequently by peers, evicting alone can not effectively reduce their replicas on the P2P network.

To examine the effectiveness of the PopCap protocol in exploiting the proxy cache, we vary the proxy cache size and investigate the system's performance, and com-

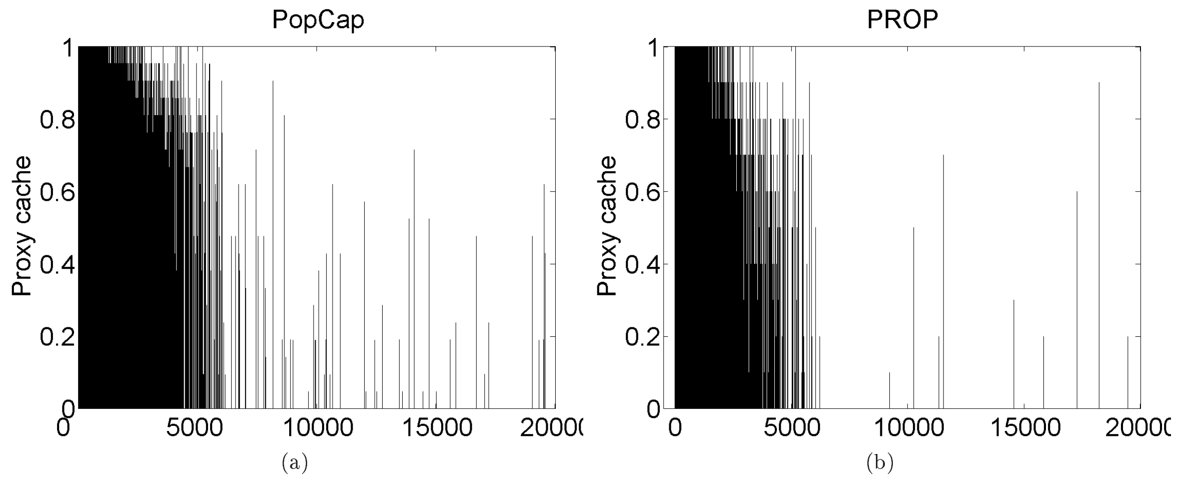


Fig. 5 Proxy caching status of the first 20,000 videos in the "sci" dataset under (a) PopCap and (b) PROP

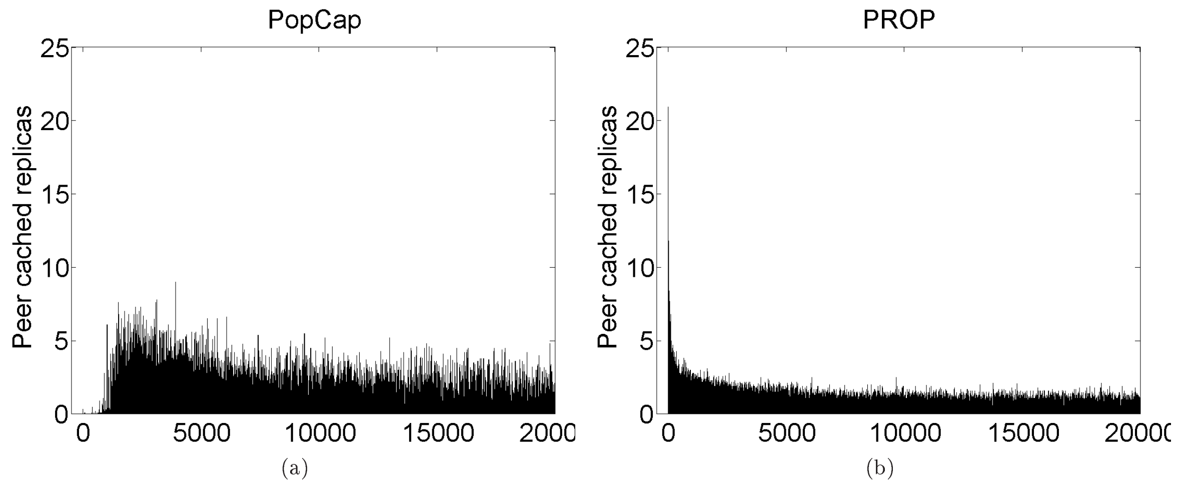


Fig. 6 Peer caching status of the first 20,000 videos in the "sci" dataset under (a) PopCap and (b) PROP

pare it with the performance of PROP. We use both the YouTube "sci" dataset and the YouTube "0316" dataset in this experiment, and for the latter one, all the 42,628 videos are used. The experimental results using the "sci" dataset are shown in Fig. 7(a) and the results using the "0316" dataset are in Fig. 7(b). From the two figures, one can see that when the proxy's cache size is small, PROP is better than PopCap. This because when the benefit of proxy caching is not significant, by using the accurate information on the number of video replicas cached by the P2P network, PROP can make a better use of the peers' cache space than PopCap. However, with the increase of the proxy's cache size, PopCap outperforms PROP. This is because PopCap makes better use of the proxy cache than PROP, and more importantly, peers under PopCap can cooperate better with the proxy by avoiding caching the videos that are likely to be cached by the proxy.

We next consider the influence of peer's cache size. In this experiment we also use the YouTube "sci" dataset and the YouTube "0316" dataset for simulation. Fig. 8 shows the traffic of the video server's uploading to the clients under varying peer cache size. Note that for letting the peer cache size as zero we mean the case that no peer-assistance is used for Internet VoD, and for the peer size as 185 seconds we mean the case that users only share their currently played videos. From Fig. 8, we can see that PopCap has a better performance than PROP, but when the peer size gets increased, the benefit is getting smaller. Comparing with the PROP's peer cache algorithm, peers under PopCap is benefit from the blocking by avoiding caching very popular videos, but PROP exploits the accurate information of the video replica number, which is assume to be available in our experiment, therefore the advantage of PopCap over PROP is diminished when the peer cache size is large enough. To validate this point we also compare PopCap with a

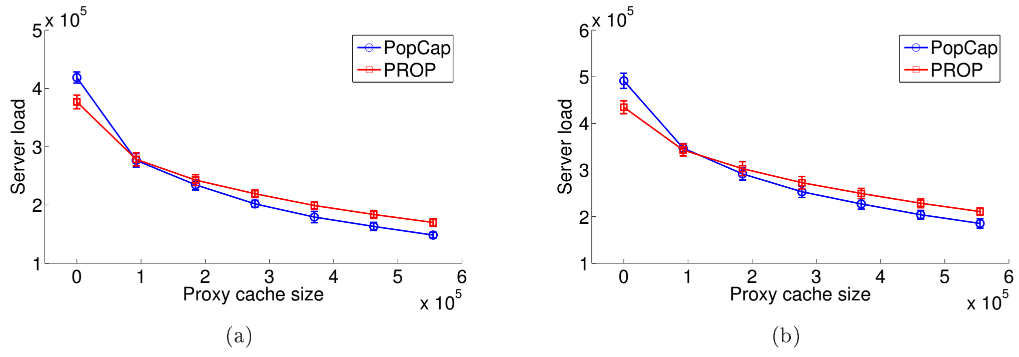


Fig. 7 Video server's uploading traffic under varying proxy cache size under PopCap and PROP, using (a) the "sci" dataset, and (b) the "0316" dataset

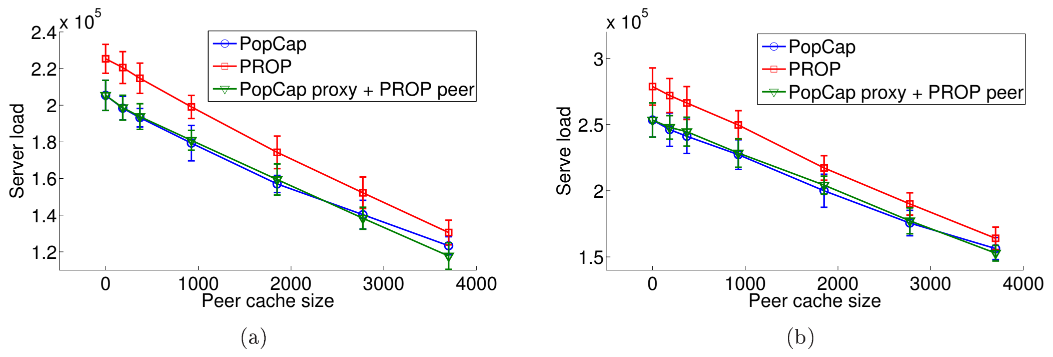


Fig. 8 Video server's uploading traffic under varying peer cache size under PopCap and PROP, using (a) the "sci" dataset, and (b) the "0316" dataset

protocol whether proxy updates its cache using the PopCap's algorithm while peers apply the PROP's algorithm while managing their caches. From Fig. 8 we can see that first of all the difference between PopCap and the new protocol is not significant, indicating that without using the costly information of the video replica number, the performance of the PopCap's peer algorithm is very close to the one of PROP. By carefully examining the figures, we can see that when the peer cache size is not very large, the PopCap's algorithm is better than the PROP's as peers under PopCap is able to avoiding caching the videos that are likely to be cached by the proxy, but when the cache is large enough, PROP's algorithm is better as the costy information of the accurate video replica number is exploited.

For a large scale information service such as Internet VoD, scalability is a very important property. To investigate how scalable the Internet VoD service is under different protocols, we vary the number of the online peers and study the system's performance. In Fig. 9, we plot the traffics on the video server and the proxy under different protocols when there are a varying number of online peers. We use the 20,000 videos in the "sci" dataset in this experiment. From Fig. 9(a) one can see that with more and more users in the VoD service, the workload on the video server gets increased, but thanks to the P2P network, the increase is not linear. Fig. 9(a) also shows that PopCap has a smaller workload on the video server than that of PROP all the time. And from Fig. 9(b) we can see that PROP also has a smaller traffic caused by proxy updates.

7.2 Effectiveness of "smart update"

In all our previous experiments we do not enable the PopCap's "smart update" mechanism for the proxy. In our last experiment we investigate the benefit of this mechanism. We use the "sci" dataset in this experiment, and to emulate the dynamic scenario of the Internet VoD service, we only have $10,000$ randomly selected videos in the system at the beginning of the experiment, and during each interval, 50 randomly selected videos from the remaining ones in the dataset are added until all the videos are added. We change the threshold for doubling the proxy update interval, i.e., the parameter of "*threshold*" in Section 5, and perform the simulation. We plot the total uploading traffic of the video server to the clients as well as the traffic on the video server caused by the proxy's updating in Fig. 10. From the figure, one can

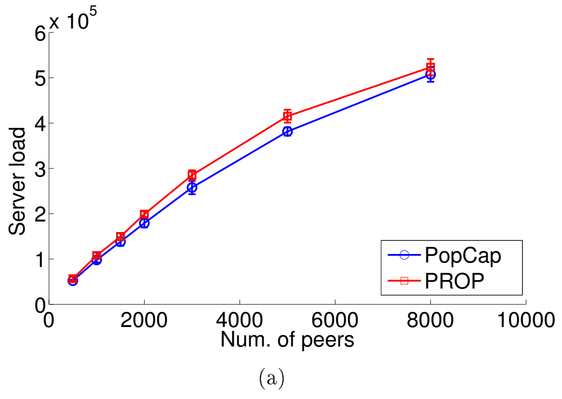


Fig. 9 (a) Server's upload traffic and (b) proxy's download traffic of peer-assisted Internet VoD using PopCap and PROP under different system size

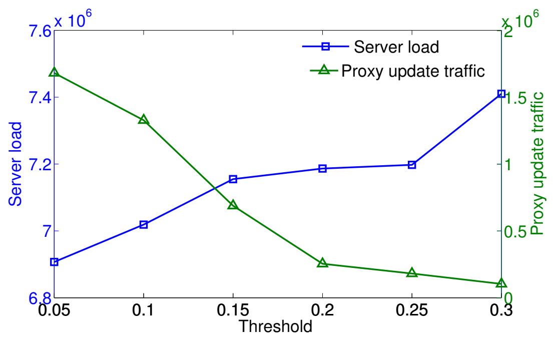


Fig. 10 Traffics caused by the video server's uploading to the clients and caused by the proxy's updating using PopCap's "smart update" mechanism under different threshold

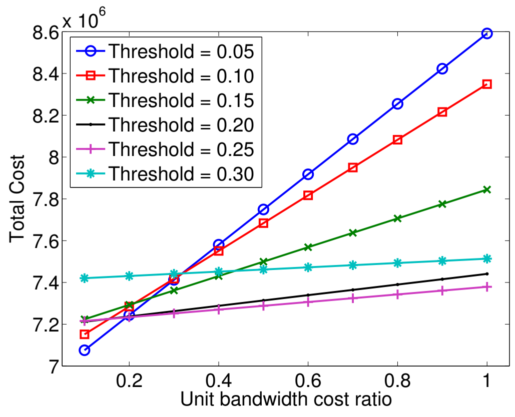


Fig. 11 Overall bandwidth cost of the video server using PopCap's "smart update" mechanism under different threshold and unit bandwidth cost ratio

see that when the threshold is increased, the video server needs to upload more to the clients as the proxy is less sensitive to the changes of the video set and its popularity, but the traffic caused by the proxy's updating is getting lighter, as the proxy updates "lazily".

Although for the requests of the ordinary clients and for the updating of the proxy, videos are downloaded from the video server, the cost for unit bandwidth may be different. For example, some ISPs apply different rates for the traffics at the peak hours and the non-peak hours. The traffic caused by the users' video requests is more likely to be at the peak hours while the proxy could update during the non-peak hours. An other example is that VSP may rent a CDN to update the proxies but users download directly from the server. When the unit bandwidth cost for the server/clients traffic and the server/proxy traffic is different, we examine the effects of the "smart update" mechanism from the economic aspect. We calculate the total bandwidth cost of the video server using the following expression

$$cost = server/clients \; + \; ratio \times server/proxy$$

where "*ratio*" is the ratio of the cost for the unit "server/clients" bandwidth and the one of the "server/proxy" bandwidth. We plot the overall cost under different thresholds applying varying ratios in Fig. 11. From the figure we can see that when the traffic cost ratio is getting larger, or proxy updating is not cheap, it is more economic beneficial to apply a larger threshold, that is, the proxy should updates more "lazily". However, note that the threshold should not be too large, as suggested by the 6th curve in the figure, where the curve is above the 4th and 5th curves under all the cost ratios. This is because for a too large threshold, the proxy somewhat fails to capture the videos' popularity dynamics.

8 Conclusion

In this paper, we consider the newly emerging Internet on-demand video streaming service and the problem of how to collaboratively use the proxy and the P2P network to cache and distribute the videos and reduce the workload on the video server. We first used two real-world datasets from YouTube to study the characteristics of the Internet VoD service. The we formally presented the video caching problem and obtained it optimal solution. Inspired by the modeling analysis, we designed a practical and inexpensive protocol named "PopCap" for the proxy and the peers in the P2P network to independently cache videos and cooperatively reduce the video server's workload. Comparing with existing solutions, PopCap requires less infrastructure support, incurs a smaller overhead, and is more easy and practical to be deployed under the Internet VoD service. Finally, through experiments based on simulation using real-world datasets, we showed that PopCap has a better performance regarding the reduction on the traffics of the video server and the proxy.

References

1. YouTube. http://www.youtube.com/.
2. Tudou.com. http://www.tudou.com/.
3. Youku. http://www.youku.com/.
4. Kevin C. Almeroth and Mostafa H. Ammar. The use of multicast delivery to provide a scalable and interactive video-on-demand service. *IEEE J. Sel. Areas Commun.*, 14(6):1110–1122, 1996.
5. Yan Huang, Tom Z. J. Fu, Dah-Ming Chiu, John C. S. Lui, and Cheng Huang. Challenges, design and analysis of a large-scale p2p-vod system. In *Proc. of ACM SIGCOMM'08*, Seattle, WA, USA., Aug. 2008.
6. YouTube serves up 100 million videos a day online. http://www.usatoday.com/tech/news/2006-07-16-youtube-views-x.htm.
7. Web could collapse as video demand soars. http://www.telegraph.co.uk/news/uknews/1584230/Web-could-collapse-as-video-demand-soars.html.
8. YouTube looks for the money clip. http://techland.blogs.fortune.cnn.com/2008/03/25/youtube-looks-for-the-money-clip/.
9. Cheng Huang, Jin Li, and Keith W. Ross. Can internet video-on-demand be profitable? In *Proc. of ACM SIGCOMM'07*, Kyoto, Japan, Aug. 2007.
10. Songqing Chen, Bo Shen, Susie Wee, and Xiaodong Zhang. Segment-based streaming media proxy: Modeling and optimization. *IEEE Trans. Multimedia*, 8(2):243–256, 2006.
11. Lei Guo, Songqing Chen, and Xiaodong Zhang. Design and evaluation of a scalable and reliable p2p assisted proxy for on-demand streaming media delivery. *IEEE Trans. Knowl. Data Eng.*, 18(5):669–682, 2006.
12. Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proc. of IMC'07*, San Diego, CA, USA, Oct. 2007.
13. MSN Video. http://video.msn.com/.
14. Xu Cheng and Jiangchuan Liu. Nettube: Exploring social networks for peer-to-peer short video sharing. In *Proc. of IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009.
15. Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic charaterization: A view from the edge. In *Proc. of IMC'07*, San Diego, CA, USA, Oct. 2007.
16. Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2cast: Peer-to-peer patching scheme for vod service. In *Proc. of WWW'03*, Budapest, Hungary, May 2003.
17. Tai T. Do, Kien A. Hua, and Mounir A. Tantaoui. P2cast: Peer-to-peer patching scheme for vod service. In *Proc. of IEEE ICC'04*, Paris, France, Jun. 2004.
18. Dan Wang and Jiangchuan Liu. A dynamic skip list-based overlay for on-demand media streaming with vcr interactions. *IEEE Trans. Parallel Distrib. Syst.*, 19(4):503–514, 2008.
19. Yi Cui, Baochun Li, and Klara Nahrstedt. oStream: Asynchronous streaming multicast in application-layer overlay networks. *IEEE J. Sel. Areas Commun.*, 22(1):91–106, 2004.
20. Ye Tian, Di Wu, and Kam-Wing Ng. A novel caching mechanism for peer-to-peer based media-on-demand streaming. *J. Syst. Archit.*, 54(1-2):55–69, 2008.
21. Alan T.S. Ip, Jiangchuan Liu, and John Chi-Shing Lui. Copacc: An architecture of cooperative proxy-client caching system for on-demand media streaming. *IEEE Trans. Parallel Distrib. Syst.*, 18(1):70–83, 2006.
22. Mohamed Hefeeda and Osama Saleh. Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Trans. Netw.*, 16(6):1447–1460, 2008.
23. Siddhartha Annapureddy, Saikat Guha, Christos Gkantsidis, Dinan Gunawardena, and Pablo Rodriguez. Exploring vod in p2p swarming systems. In *Proc. of IEEE INFOCOM'07*, Anchorage, AK, USA, May 2007.
24. Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. Understanding user behavior in large-scale video-on-demand systems. In *Proc. of EuroSys'06*, Leuven, Belgium, Apr. 2006.
25. Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, and Xiaodong Zhang. The stretched exponential distribution of internet media access patterns. In *Proc. of ACM Symposium on Principles of Distributed Computing (PODC'08)*, Toronto, Canada, Aug. 2008.
26. Haitham Hindi. A tutorial on convex optimization. In *Proc. of 2004 American Control Conference*, Boston, MA, USA, Jun. 2004.
27. Xinyan Zhang, Jiangchuan Liu, Bo Li, and T.-S. Peter Yum. Donet/coolstreaming: A data-driven overlay network for live media streaming. In *Proc. of IEEE INFOCOM'05*, Miami, FL, USA, Mar. 2005.
28. Donald Gross and Carl M. Harris. *Fundamentals of queueing theory (2nd ed.).* John Wiley & Sons, Inc., New York, NY, USA, 1985.