

# Cloud Download: Using Cloud Utilities to Achieve High-quality Content Distribution for Unpopular Videos \*

Yan Huang<sup>1</sup>, Zhenhua Li<sup>2\*</sup>, Gang Liu<sup>1</sup>, and Yafei Dai<sup>2</sup>

<sup>1</sup> Tencent Research, Shanghai, China

<sup>2</sup> EECS, Peking University, Beijing, China

{galehuang, lizhenhua1983 \*, ganghust, daiyafei} @gmail.com

## ABSTRACT

Video content distribution dominates the Internet traffic. The state-of-the-art techniques generally work well in distributing popular videos, but do not provide satisfactory content distribution service for *unpopular videos* due to low data health or low data transfer rate. In recent years, the worldwide deployment of cloud utilities provides us with a novel perspective to consider the above problem. We propose and implement the *cloud download* scheme, which achieves high-quality video content distribution by using cloud utilities to *guarantee the data health and enhance the data transfer rate*. Specifically, a user sends his video request to the cloud which subsequently downloads the video from the Internet and stores it in the cloud cache. Then the user can usually retrieve his requested video (whether popular or unpopular) from the cloud with high data rate in any place at any time, via the intra-cloud data transfer acceleration. Running logs of our real deployed commercial system (named VideoCloud) confirm the effectiveness and efficiency of cloud download. The users' average data transfer rate of unpopular videos exceeds 1.6 Mbps, and over 80% of the data transfer rates are more than 300 Kbps which is the basic playback rate of online videos. Our study provides practical experiences and valuable heuristics for making use of cloud utilities to achieve efficient Internet services.

## Categories and Subject Descriptors

C.2.4 [Distributed Systems]: Distributed applications

## General Terms

Design, Measurement, Performance

---

\*Area chair: Surender Chandra.

This paper is a joint work of Tencent Research and Peking University. The first two organizations/authors have made their unique and equally important contributions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.

Copyright 2011 ACM 978-1-4503-0616-4/11/11 ...\$10.00.

## 1. INTRODUCTION

Video content distribution dominates the Internet traffic. A recent Cisco report [1] says that nearly 90% of all the consumer IP traffic is expected to consist of video content distribution, including web video like YouTube [2], P2P (peer-to-peer) video like BitTorrent [3] and PPLive [4], and so on in 2012. Therefore, achieving *high-quality* video content distribution over the Internet is of great significance for both academia and industry. Here the meaning of *high-quality* is two-fold: *high data health* and *high data transfer rate*. The metric *data health* is initially used in the BitTorrent protocol [3]. It denotes the number of available full copies of the shared file in a BitTorrent swarm. For example, the data health 1.0 means one full copy is available, and a swarm with data health less than 1.0 is called an *unhealthy* swarm because no peer in the swarm can get a full copy. In this paper, we keep using *data health* to represent the data redundancy level of a video file. High data health implies that the user is able to obtain a full copy of the video, and high data transfer rate enables the advanced function of online video streaming (including live streaming [5] and on-demand streaming [6]).

The state-of-the-art techniques of video content distribution mainly include CDN (content distribution network like Akamai [7]) and P2P. CDN is the traditional technique that optimizes the performance of Internet content distribution by strategically deploying edge servers at multiple locations (often over multiple ISP networks). These edge servers cooperate with each other by replicating or moving data according to data popularity and server load. An end user usually obtains a copy of data from a nearby edge server, so that the data transfer rate is greatly enhanced and the load on the original data source is reduced. However, for the sake of limited storage and bandwidth capacity, it is not cost-effective for CDN to replicate unpopular videos to edge servers [8], considering that there are many more unpopular videos than popular videos over the Internet. Moreover, CDN is a charged facility that only serves the content providers who have paid, rather than a public utility of the Internet. Consequently, it is impractical to count on CDN for efficient distribution of unpopular videos.

Different from CDN, P2P content distribution mainly relies on the unstable but numerous end users to form peer-to-peer data swarms [3, 9], where data is directly exchanged between neighboring peers. The real strength of P2P shows when a popular video is distributed, because a popular video is shared by a number of peers and more peers usually imply higher data health and higher degree of download paral-

lelism, which further lead to higher data transfer rate. As to an unpopular video, it is often difficult to find a corresponding peer swarm. Even if the peer swarm exists, the few peers are unlikely to have high data health or high data transfer rate, and thus each peer has to stay online for long hours to wait for the download completion — a tedious process of low energy efficiency. In a word, although CDN and P2P generally work well in distributing popular videos, neither of them is able to provide satisfactory content distribution service for unpopular videos, due to low data health or low data transfer rate.

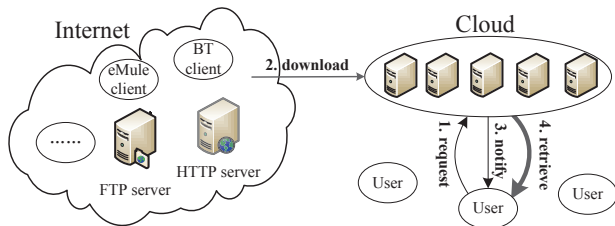


Figure 1: Principle of cloud download.

In recent years, the worldwide deployment of cloud utilities [10, 11] provides us with a novel perspective to consider the above problem. In this paper, we propose and implement the *cloud download* scheme, which achieves high-quality video content distribution by using cloud utilities to *guarantee the data health* and *enhance the data transfer rate*. The principle of cloud download is depicted in Figure 1. Firstly, a user sends his video request to the cloud (see Arrow 1 in Figure 1). The video request contains a file link which can be an HTTP/FTP link, a BitTorrent [3]/eMule [9] link, and so on<sup>1</sup>. Subsequently, the cloud *downloads* the requested video from the file link and stores it in the cloud cache (see Arrow 2 in Figure 1) to guarantee the data health of the video  $\geq 1.0$  (each video also has a duplicate in the cloud cache for redundancy). Then the cloud notifies the user (see Arrow 3 in Figure 1) and the user can usually *retrieve*<sup>2</sup> his requested video (whether popular or unpopular) from the cloud with high data rate (see Arrow 4 in Figure 1) in any place at any time, via the intra-cloud data transfer acceleration (which will be addressed in detail in Section 3.2). In practice, the cloud does not need to notify the user when his requested video is available. Instead, the user actively checks the download progress by himself and takes corresponding actions. That is to say, Arrow 3 in Figure 1 can also be replaced by “3. check”.

Besides, an important derivative advantage brought by cloud download is the *user-side energy efficiency*. An Internet user often has to keep his computer (and network interface card) powered-on for long hours (says  $t_1$ ) only to download an unpopular video [12]. During this process, a lot of energy is wasted because the major components (including CPU, memory, disk, etc.) keep working but are not

<sup>1</sup>The cloud does not accept keywords as the video request, because one keyword or a combination of several keywords may correspond to multiple videos with different contents in the Internet.

<sup>2</sup>Here we use *retrieve* to distinguish with *download*. In this paper, *download* means the cloud obtains data from the Internet while *retrieve* means the user obtains data from the cloud.

efficiently used. Cloud download uses the cloud servers to download the requested videos, so its users do not need to be online; in fact, they do not need to open their computers at all — this is the reason why we also call “cloud download” as “offline download”. When the requested video is available in the cloud, the user can usually retrieve it in short time (says  $t_2$ ). Thus, cloud download significantly reduces the user-side energy cost (by the ratio  $\frac{t_1-t_2}{t_1}$ ). The detailed energy efficiency analysis will be presented in Section 4.5.

The only drawback of cloud download lies in that for some videos, the user must wait for the cloud to download it and thus he cannot view the video at once. The abovementioned waiting time is denoted as the *view startup delay*. Note that CDN and P2P also have their view startup delay to buffer a certain amount of data for a smooth view startup of the video (e.g., buffer the first several minutes’ data for a 100-minute video). This drawback is effectively alleviated by the *implicit and secure data reuse among the users* of cloud download. For each video, the cloud only downloads it from the Internet when it is requested for the first time, and the subsequent requests are directly satisfied by using the cached copy with very low view startup delay (except when the cached copy is replaced). Such data reuse is more secure than the explicit data reuse among P2P users (which is susceptible to the Sybil attack [13], Eclipse attack [14], DHT attack [15], and so on), because it is completely handled by the cloud and is oblivious to users. According to the real-system performance, the data reuse rate among the users reaches 87%, indicating that most video requests are satisfied instantly with very low view startup delay.

Since June 2010, we have deployed a large-scale commercial cloud download system (named VideoCloud [16]) on top of the QQcyclone platform [17], using a “micro” data center composed of 426 commodity servers. VideoCloud has attracted over 6 million registered users and supports most of the mainstream content distribution protocols like HTTP, FTP, BitTorrent, eMule, and so on. Currently, VideoCloud receives over 0.2 million video requests sent from around 50000 users per day, and most of the requests are issued for unpopular videos. The data center of VideoCloud has been deployed across four major ISPs in China and is planned to cover more domains. The user’s monetary cost for cloud download service differs from Amazon S3 [18] and Microsoft Azure [19], but is more similar to Dropbox [20]. Specifically, the user of VideoCloud is charged according to his cloud storage capacity and regardless of the bandwidth consumed. Any registered user is allocated with 5-GB free storage, and extra storage is charged in unit of 5 GB.

The system running logs confirm the effectiveness and efficiency of cloud download. For example, the users’ average data transfer rate of unpopular videos exceeds 1.6 Mbps, and over 80% of the data transfer rates are more than 300 Kbps which is the basic playback rate of online videos [2, 21]. In comparison, when the *common download method* is used, the average data transfer rate is merely 0.57 Mbps, and 70% of the data transfer rates are less than 300 Kbps. The common download method denotes the common way in which a user downloads video content from the Internet, e.g., using a web browser or a P2P client software to download a video. Besides, compared with the common download method, cloud download reduces the user-side energy cost by around 92%.

To sum up, our contributions in this paper are as follows:

1. We analyze the state-of-the-art techniques of video content distribution (i.e., CDN and P2P) and discover that neither of them is able to provide satisfactory content distribution service for unpopular videos. Driven by this problem, we propose and implement the novel cloud download scheme, which achieves high-quality video content distribution by using cloud utilities to guarantee the data health and enhance the data transfer rate. The only drawback of cloud download, i.e., the view startup delay, is effectively alleviated by the implicit and secure data reuse among the users.
2. We have deployed VideoCloud, a large-scale commercial cloud download system. And we present its system architecture and design techniques, as well as our observations, analysis, and insights. Our study provides practical experiences and valuable heuristics for making use of cloud utilities to achieve efficient Internet services.
3. We evaluate the performance of VideoCloud via its running logs using three major metrics: data transfer rate, view startup delay, and energy efficiency. The evaluation results confirm the effectiveness and efficiency of cloud download.

## 2. RELATED WORK

As the backbone bandwidth and end user access bandwidth of the Internet continuously increase, people have been putting forward higher and higher demand on video content distribution. The pixel resolution evolves from CIF (common intermediate format, 352\*288), VGA (video graphics array, 640\*480), to HD (high-definition, 1280\*720), and the playback mode evolves from view-after-download to view-as-download. In the past 15 years, high-quality video content distribution has been a very hot research topic in both industry and academia [22], in particular the state-of-the-art techniques: CDN and P2P. The advantages and disadvantages of CDN, P2P, and cloud download have been addressed in the previous section. In recent years, some researchers recognize that both CDN and P2P have their limitations and thus propose novel schemes by combining or optimizing CDN and P2P.

On top of ChinaCache [23], the biggest commercial CDN service provider in China, Yin et al. developed a *hybrid CDN-P2P* live streaming system (named LiveSky [24]) to incorporate the strengths on both sides: the scalability of P2P, and the quality control capability and reliability of CDN. Although the hybrid CDN-P2P architecture inevitably brings extra deployment complication and maintenance cost, LiveSky achieves lower burden on the CDN network and higher working efficiency of peer streaming for popular videos. However, for unpopular files, it is difficult and inefficient for the few users to form a peer swarm, and then the performance of LiveSky will be like that of a common CDN.

Wu et al. [25] refocused on the importance of servers in P2P streaming systems and thus proposed the *server-assisted P2P* streaming scheme. Via a 7-month running trace of a commercial P2P live streaming system, named UUSee [26], they found that the total available bandwidth of the 150 streaming servers could not meet the increasing demand of download bandwidth from hundreds of channels (a channel can be seen as a peer swarm), although the total upload bandwidth of peers also increased with download

demand. So they proposed an allocation algorithm of server bandwidth, named Ration, which could proactively predict the minimum server bandwidth demand of each channel, based on historical information. Therefore, each channel can be guaranteed with desirable streaming quality. Ration has the similar shortcoming as LiveSky because every channel of UUSee can be seen as a very popular video. For unpopular videos, Ration would work like the traditional client-server video streaming scheme.

Recent years have seen a strong trend in *extremely high-quality* video content distribution like online HDTV (high-definition TV) [27] and cinematic-quality VoD (video-on-demand) [28]. Undoubtedly, extremely high-quality videos can provide Internet users with wonderful viewing experiences; however, the content distribution of extremely high-quality videos (in particular those unpopular videos without powerful CDN support) over the highly dynamic Internet environments has still been a very challenging problem. The latest practical solution may be Novasky [28], an operational “P2P storage cloud” for on-demand streaming of cinematic-quality videos over a high-bandwidth network (i.e., the campus network of Tsinghua University). The limitation of Novasky is obvious: it works on a high-bandwidth campus network which is much more stable and homogeneous than the real Internet. Cloud download uses cloud utilities to guarantee the data health of videos and meanwhile enhance the data transfer rate to quite high and stable. Thus, cloud download can be taken as a promising solution to the future extremely high-quality video content distribution over the Internet.

## 3. SYSTEM DESIGN

In this section, we present the system architecture and design techniques of VideoCloud, as well as our observations, analysis, and insights.

### 3.1 System Overview

As depicted in Figure 2, the architecture of VideoCloud is mainly composed of five building blocks: 1) *ISP Proxies*, 2) *Task Manager*, 3) *Task Dispatcher*, 4) *Downloaders*, and 5) *Cloud Cache*. The detailed hardware composition of each building block is listed in Table 1. Note that all the information of Memory, Storage, and Bandwidth refers to one server. We do not list the CPU information since the performance of CPU is not important for our system which is data (bandwidth and storage) intensive rather than computation intensive. In total, the system utilizes 426 commodity servers, including 300 chunk servers making up a 600-TB cloud cache, 80 download servers with 26 Gbps of Internet bandwidth, and 33 upload servers with 20 Gbps of Internet bandwidth. Every server runs the Suse Linux v10.1 operating system. Now we describe the organization and working process of the system by following the message and data flows of a typical cloud download task.

The user of VideoCloud should have installed its client software [16]. The client software is able to recognize which ISP the user locates at from the user’s IP address (with the help of an embedded and periodically updated IP-ISP mapping file), so the video request is firstly sent to the corresponding ISP Proxy by the client software (see Arrow 1 in Figure 2). Each ISP Proxy maintains a *request queue* to control the number of requests sent to the Task Manager (see Arrow 2 in Figure 2), so that the Task Manager is resilient

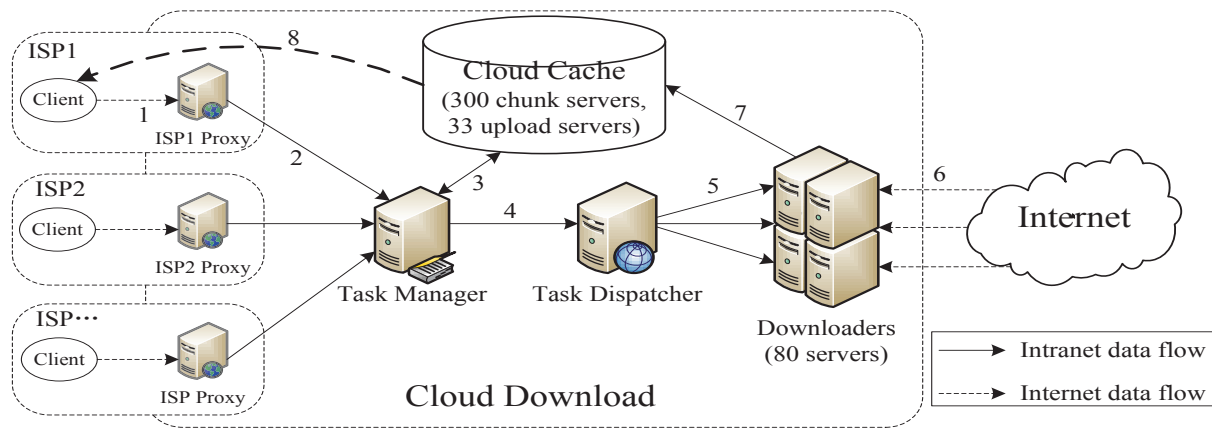


Figure 2: System architecture of cloud download.

Table 1: Hardware composition of VideoCloud.

Building Block	Number of servers	Memory	Storage	Bandwidth
ISP Proxy	4	8 GB	250 GB	1 Gbps (Intranet), 0.3 Gbps (Internet)
Task Manager	4	8 GB	250 GB	1 Gbps (Intranet)
Task Dispatcher	3	8 GB	460 GB	1 Gbps (Intranet)
Downloaders	80	8 GB	460 GB	1 Gbps (Intranet), ~0.325 Gbps (Internet)
Cloud Cache	300 chunk servers, 33 upload servers, and 2 index servers	8 GB	4 TB (chunk server), 250 GB (upload server)	1 Gbps (Intranet), ~0.6 Gbps (Internet)

to request upsurge in any ISP. Presently, VideoCloud has set four ISP Proxies in the four major ISPs in China: China Telecom [29], China Unicom [30], China Mobile [31], and CERNET (China Education and Research Network) [32]. If the user does not locate at any of the four major ISPs (such users take a small portion), his video request is sent to a random one of the four ISP Proxies. We plan to set more ISP Proxies in more ISPs in the future.

On receiving a video request, the Task Manager firstly checks whether the requested video has a copy in the Cloud Cache (see Arrow 3 in Figure 2). If the video request is a BitTorrent/eMule link, the Task Manager examines whether the Cloud Cache contains a video which has the same hash code with that contained in the BitTorrent/eMule link<sup>3</sup>. Otherwise, the Task Manager directly examines whether the file link is repeated in the Cloud Cache. If the requested video actually has a copy, the user can directly and instantly retrieve the video from the Cloud Cache (see Arrow 8 in Figure 2). Otherwise, the Task Manager forwards the video request to the Task Dispatcher (see Arrow 4 in Figure 2).

The Task Dispatcher assigns the video request to one server (called a “downloader”) in the Downloaders for data downloading (see Arrow 5 in Figure 2). For example, if the video request is a P2P request, the assigned downloader will act like a common peer to join the corresponding peer swarm. The Task Dispatcher is mainly responsible for balancing the bandwidth loads of the 80 downloaders. The number of downloaders (80) is empirically determined to guarantee that the total download bandwidth (26 Gbps) ex-

ceeds the peak load of download tasks (nearly 20 Gbps till now). Each downloader executes multiple download tasks in parallel in order to fully utilize its download bandwidth (around 0.325 Gbps) (see Arrow 6 in Figure 2), and the Task Dispatcher always assigns a newly incoming video request to the downloader which has the lowest download data rate.

As long as the downloader accomplishes a download task, it computes the hash code of the downloaded video and attempts to store the video in the Cloud Cache (see Arrow 7 in Figure 2). The downloader firstly checks whether the video has a copy in the Cloud Cache (using the hash code). If the video is repeated, the downloader simply discards it. Otherwise, the downloader checks whether the Cloud Cache has enough unused space to store the new video. If the Cloud Cache does not have enough unused space, it deletes some cached videos to get enough unused space to store the new video. The concrete cache architecture, cache capacity planning, and cache replacement strategy will be investigated in Section 3.2, Section 3.4, and Section 3.5, respectively.

When the abovementioned video store process is finished, the user can usually retrieve the requested video from the Cloud Cache (see Arrow 8 in Figure 2) in any place at any time. Since the user’s ISP information can be acquired from his video retrieve message (see Arrow 4 in Figure 1), the Cloud Cache takes advantage of the intra-cloud ISP-aware data upload technique to restrict the retrieve data flow within the same ISP as the user’s, so as to enhance the data transfer rate and avoid the inter-ISP traffic cost.

### 3.2 Data Transfer Acceleration

A critical problem of cloud download is how to accelerate the data transfer (i.e., retrieve) process so that the user can

<sup>3</sup>A BitTorrent/eMule link contains the hash code of its affiliated file in itself, while an HTTP/FTP link does not.

obtain his requested video (whether popular or unpopular) from the cloud with high data rate in any place at any time. Considering that the cross-ISP data transfer performance degrades seriously and the inter-ISP traffic cost is often expensive [33, 34], we solve this problem via the intra-cloud ISP-aware data upload technique. Since the user’s real-time ISP information can be acquired from his video retrieve message, the Cloud Cache takes advantage of its ISP-aware upload servers to restrict the retrieve data flow within the same ISP as the user’s, so as to enhance the retrieve data rate and avoid inter-ISP traffic cost. Specifically, as shown in Figure 3, the Cloud Cache consists of 300 chunk servers, 33 upload servers, and 2 index servers which are connected by a DCN (data center network). A specific number of upload servers are placed in each ISP, proportional to the data traffic volume in each ISP. The video requests come from tens of ISPs, but most of them come from the four major ISPs in China: China Telecom [29], China Unicom [30], China Mobile [31], and CERNET [32]. Figure 4 demonstrates the number of users, the number of video requests, and the data traffic volume in each ISP in one day. Thereby, the Cloud Cache places 21, 10, 1, and 1 upload servers in the four major ISPs respectively at the moment. If the user locates at other ISPs, we randomly choose an upload server to transfer the corresponding data.

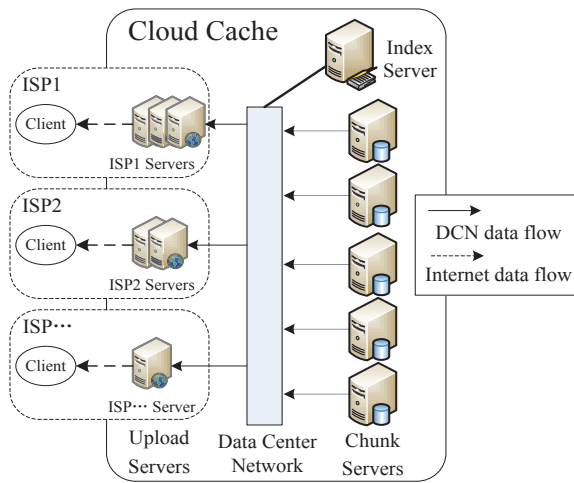


Figure 3: Architecture of the Cloud Cache.

Every video is segmented into chunks of equal size to be stored in the chunk servers, and every chunk has a duplicate for redundancy, so the 300 chunk servers can accommodate a total of  $\frac{300 \times 4 \text{ TB}}{2} = 600 \text{ TB}$  unique data. In order to achieve load balance and exploit the chunk-correlation in the same file, all the chunks of a video are stored together into the chunk server with the biggest available storage capacity. The duplicate chunks of a video must be stored in another chunk server. There exists an index server (as well as a backup index server) which maintains the metadata of chunks for chunk search and validation. The metadata is a list of  $n$ -tuples like  $\langle \text{file hash code}, \text{file link}, \text{number of chunks}, \text{physical location of the first chunk}, \text{physical location of the first duplicate chunk} \rangle$ .

The DCN in the Cloud Cache adopts the traditional three-tier structure to organize the switches, composed of a core tier in the root of the tree, an aggregation tier in the

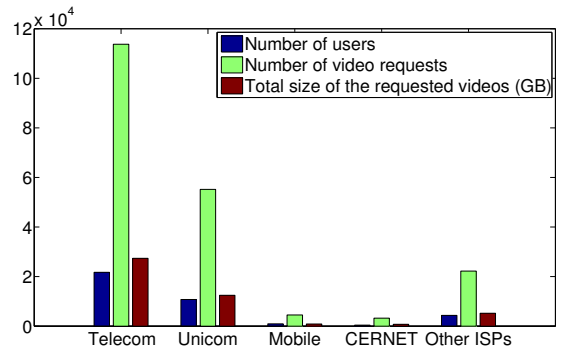


Figure 4: Number of users, number of video requests, and data traffic volume in each ISP in one day.

middle, and an edge tier at the leaves of the tree. The index servers, chunk servers, upload servers, and downloaders are connected to the switches at the edge tier. Suppose a user  $A$  locating at ISP1 wants to retrieve a video  $f$  stored in a chunk server  $S$ , and the Cloud Cache has placed 10 upload servers ( $U_1, U_2, \dots, U_{10}$ ) in ISP1. The chunks of  $f$  are firstly transferred from  $S$  to a random upload server (says  $U_4$ ) in ISP1, and then transferred from  $U_4$  to the user  $A$ . The transfer process is not store-and-forward but pass-through: as soon as  $U_4$  gets a complete chunk of  $f$ ,  $U_4$  transfers the chunk to  $A$ .  $f$  would not be cached in  $U_4$  because the intra-cloud end-to-end bandwidth is quite high (1 Gbps) and we do not need to make things more complicated than necessary.

### 3.3 Download Success Rate Improvement

Although cloud download guarantees the data health of the requested video ( $\geq 1.0$ ) after the video is downloaded by the cloud, it cannot guarantee to download every requested video successfully. In this subsection, we construct a simple model to analyze the download success rates of cloud download and *common download* (the data reuse among users is not considered). *Common download* is the common way in which a user downloads video content from the Internet. Considering the highly dynamic and complicated Internet environments, the target of our model is to illustrate the long-term expected trend, rather than the accurate prediction in a short period.

Whether we use common download or cloud download, the download success rate of a requested HTTP/FTP file only depends on the accessibility of the file link (1-accessible or 0-inaccessible). According to our measurements of the accessibility of numerous HTTP/FTP file links submitted to VideoCloud, the average download success rate of HTTP/FTP files is:

$$R_1 = R'_1 = 0.414.$$

The download success rate of a requested P2P file is much more complicated because the accessibility of a peer is a stochastic variable. Consider a P2P swarm consisting of  $n$  peers sharing a file  $f$ . The time limit is  $T$  hours, that is to say, each peer must draw his conclusion in  $T$  hours: 1-download success or 0-download failure. In average, each peer stays online for  $t$  hours in the time limit ( $T$  hours) and the data health of a single peer is  $h$  ( $h < 1.0$ ). For a common peer  $P$ , during the  $t$  hours when  $P$  is online,  $P$  is expected

to encounter  $\frac{n-t}{T}$  online peers ( $n$  is the average number of peers in a peer swarm), assuming the online time of each peer is independent. Since the data health of a single peer is  $h$ , the download success rate of  $P$  is expected to be:

$$R_2 = 1 - (1 - h)^{\frac{n-t}{T}}.$$

Cloud download uses a stable peer  $P'$  (in fact,  $P'$  is a cloud server) to join in the P2P swarm, so the online duration  $t$  for  $P'$  is  $t = T$ . Then the download success rate of  $P'$  is expected to be:

$$R'_2 = 1 - (1 - h)^n.$$

Among all the video requests, let  $\alpha$  denote the fraction of HTTP/FTP file requests, and  $\beta$  denote the fraction of P2P file requests. Thus, the overall download success rates of common download and cloud download are expected to be:

$$R = \alpha \cdot R_1 + \beta \cdot R_2, \text{ and } R' = \alpha \cdot R'_1 + \beta \cdot R'_2.$$

Our 17-day measurements of VideoCloud show that  $h = 0.4$ ,  $n = 5.4$ ,  $\frac{t}{T} = 0.12$ ,  $\alpha = 27.4\%$ , and  $\beta = 72.6\%$ . As a result,

$$R = 31.8\%, \text{ and } R' = 79.3\%.$$

The expected download success rates ( $R$  and  $R'$ ) are depicted in Figure 5 to compare with the real performance of VideoCloud. The real-system average download success rate is 81.4%.

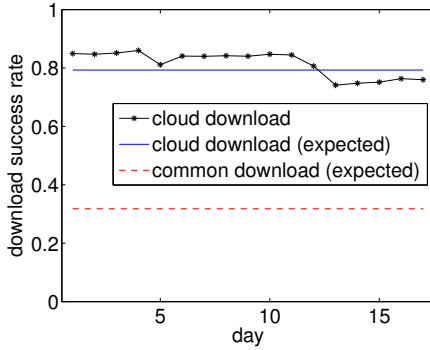


Figure 5: Download success rate in each day.

### 3.4 Cache Capacity Planning

The biggest hardware and energy cost of the VideoCloud system comes from the Cloud Cache, in particular the 300 chunk servers making up a 600-TB cloud cache. In this subsection, we address the reason why the cache capacity is planned as 600 TB.

VideoCloud has over 6 million registered users. It is impossible to allocate each user with unlimited cloud cache capacity. Instead, each registered user is allocated with a 5-GB free cloud cache at the moment, because 5 GB is around the size of a common TV play series<sup>4</sup>. Given that the average size of requested videos is 379 MB, we regard a cloud cache which accommodates more than 13 videos ( $5 \text{ GB} / 379 \text{ MB} = 13.5$ ) as basically enough to satisfy a common user's

<sup>4</sup>Amazon Cloud Drive [35] also sets 5 GB as the basic cloud storage capacity provided to its users.

requirement. Extra storage is charged in unit of 5 GB. According to our statistics, almost all the users of VideoCloud are 5-GB free users, which is quite similar to the situation of Dropbox [20]. Consequently, over 6 million registered users need more than 29000 TB of cache capacity *in the worst case*. The *worst case* happens under the following three conditions: 1) Every user fully utilizes its 5-GB cloud cache<sup>5</sup>; 2) There is no data reuse among the users; and 3) The data in a user's cloud cache is stored *forever* unless it is retrieved by the user (this condition will be replaced by a more practical and economical condition later on). It is difficult to predict the extent of the first and second conditions because they depend on users' behaviors and may vary enormously as time goes by. Therefore, to change the third condition is our only choice. The third condition is replaced by a practical and economical condition: 3) The data in a user's cloud cache is stored *for one week* unless it is retrieved by the user. The store period (one week) is an empirical parameter according to our operating experiences of VideoCloud, i.e., almost all the users (over 95%) retrieve their requested videos within one week after the requested video is available. Currently, VideoCloud receives around 0.22 million video requests per day and the average size of requested videos is 379 MB, so the total cloud cache capacity *in the worst case* should be:

$$C = 379 \text{ MB} \times 0.22M \times 7 = 584 \text{ TB}.$$

To cope with the fluctuations in the daily number of video requests, the total cloud cache capacity is planned as  $C' = 600 \text{ TB}$ , slightly larger than  $C = 584 \text{ TB}$ . Via the 600-TB cloud cache, the data reuse rate among the users reaches 87%, indicating that most video requests are satisfied instantly (i.e., with very low view startup delay).

### 3.5 Cache Replacement Strategy

Although the current cloud cache capacity (600 TB) can well handle the current daily number of video requests (around 0.22 million), it is quite likely that the number of video requests will be much higher than 0.22 million in some day. For example, if the daily number of video requests increases to 2.2 million, what shall we do? Obviously, it is impossible for us to construct a 6000-TB cloud cache made up of 3000 chunk servers — the hardware and energy cost are far beyond our affordability. As a result, some data must be replaced to make efficient utilization of the limited cloud cache capacity, where the cache replacement strategy plays a critical role. In this subsection, we investigate the performance of the most commonly used cache replacement strategies, i.e., FIFO (first in first out), LRU (least recently used), and LFU (least frequently used) via real-trace driven simulations. The trace is a 17-day system log (refer to Section 4.1 for detailed information) of VideoCloud. The metrics are two-fold: 1) *cache hit rate*; and 2) *size of replaced data*, which illustrates the I/O cost of chunk servers. We hope the cache hit rate to be as high as possible, while the size of replaced data to be as small as possible.

As shown in Figure 6 and Figure 7, among the three cache replacement strategies, FIFO performs the worst, and LFU performs the best to achieve the highest cache hit rate and the smallest size of replaced data. The cache capacity is set to 70 TB, and using other cache capacities rather than

<sup>5</sup>Since almost all the users of VideoCloud are 5-GB free users at the moment, we regard every user as a 5-GB free user for computation convenience.

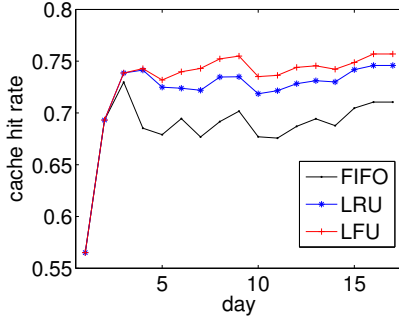


Figure 6: Cache hit rate.

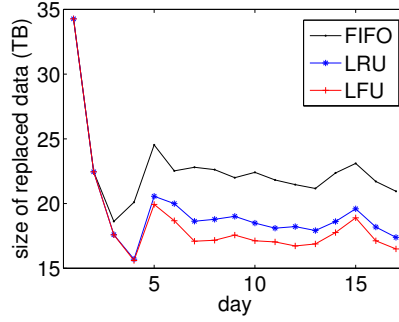


Figure 7: Size of replaced data.

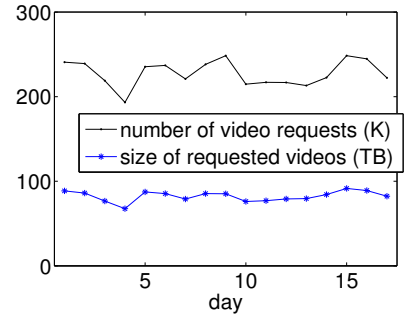


Figure 8: Daily statistics.

70 TB generates the similar results. Therefore, the Cloud Cache of VideoCloud adopts the LFU cache replacement strategy for future scalability. Delving more deeply, LFU has been recognized to be especially suitable for a system where the popularity of data objects does not change very much over a specific time period (one day or one week) [36]. VideoCloud has the abovementioned property because most of the data objects are unpopular videos whose popularity hardly changes much.

## 4. PERFORMANCE EVALUATION

### 4.1 Dataset

We use the complete running log of the VideoCloud system in 17 days, from January 1, 2011 to January 17, 2011 to evaluate the performance of cloud download. The log includes the performance information of around 3.87 million video requests, involving around one million unique videos. Most of the videos are .rmvb (around 40%) and .avi files (around 20%). 27.4% of the requested videos are HTTP/FTP files, and the remaining are BitTorrent/eMule files. The total size of the requested videos is 1400 TB, and the total size of the unique videos is 280 TB. The daily statistics are plotted in Figure 8. For each video request, we record its *user id*, *file link*, *file hash code*, *file type*, *file size*, *video request time*, *download duration time* (of the cloud downloader), *retrieve duration time* (of the users), *cache hit status* (1-hit, 0-miss), *download success status* (1-success, 0-failure), and so on.

Figure 9 indicates that the popularity distribution of the requested videos in 17 days is highly skewed, approximately following the Zipf model [37]. Let  $x$  denote the popularity ranking of a video file, and let  $y$  denote the popularity of the file. Then we have the following fitting equation:

$$\log(y) = -a \cdot \log(x) + b,$$

which is equal to

$$y = 10^b \cdot x^{-a},$$

where  $a = 0.8464$  and  $b = 11.3966$ . More in detail, 97.1% of the videos receive less than one request per day while merely 0.09% of the videos receive more than ten requests per day (we empirically use the indicator “ten requests per day” as the boundary between popular and unpopular videos), demonstrating that almost all the requested videos in VideoCloud are unpopular. Besides, around 83% of the requests are issued for unpopular videos. As mentioned

in Section 1, our cloud download scheme aims to provide efficient content distribution service for unpopular videos, so we only evaluate the performance corresponding to unpopular videos in the following subsections.

### 4.2 Metrics

We use three major metrics as follows to evaluate the performance of a cloud download system.

1. *Data transfer rate* denotes the data rate when a user retrieves his requested video from the cloud. Specifically, data transfer rate =  $\frac{\text{file size}}{\text{retrieve duration time}}$ .
2. *View startup delay* denotes how long a user must wait for the cloud to download his requested video. For a video request, if the cache hit status is 1-hit, the view startup delay is taken as 0; otherwise, the view startup delay is regarded as the download duration time of the cloud downloader.
3. *Energy efficiency* denotes how much energy is saved by using cloud download, compared with the common download method. In particular, we consider the *user-side* energy efficiency and the *overall* energy efficiency, respectively.

### 4.3 Data Transfer Rate

Since the data transfer rate is computed by dividing the file size by the retrieve duration time, in this subsection we firstly present the CDF of file size (in Figure 10), the CDF of retrieve duration time (in Figure 11) and the CDF of data transfer rate (in Figure 12), and then try to discover their relations.

The average file size is 379 MB, which is close to the common size of a 100-minute video, given the fact that around 60% of the requested videos are .rmvb and .avi files. From Figure 10, we find that 16% of the files are smaller than 8MB, most of which are demo videos, pictures, and documents. The reason lies in that many video content providers like to attach the introductions, snapshots, or advertisements to the videos.

The average retrieve duration time is 32 minutes. From Figure 11, we see that 73% of the files are retrieved in less than 32 minutes, and 93% of the files are retrieved in less than 100 minutes so that if the file is a 100-minute video, it can be viewed with high playback continuity (the user can choose either mode: view-as-download or view-after-download).

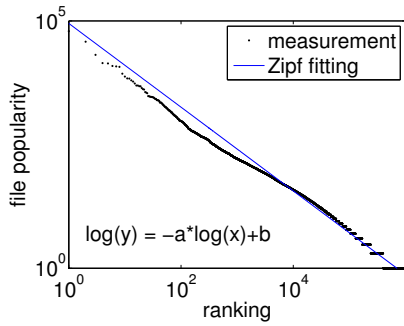


Figure 9: Popularity distribution of requested videos.

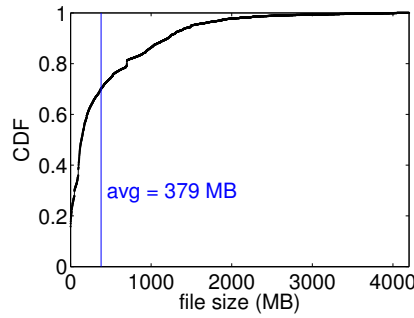


Figure 10: CDF of file size.

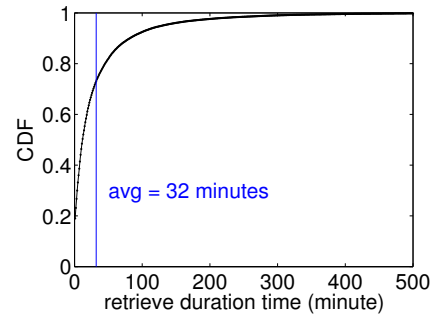


Figure 11: CDF of retrieve duration time.

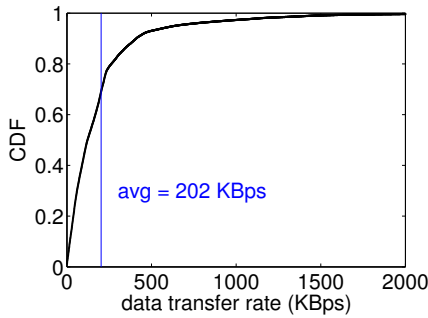


Figure 12: CDF of data transfer rate.

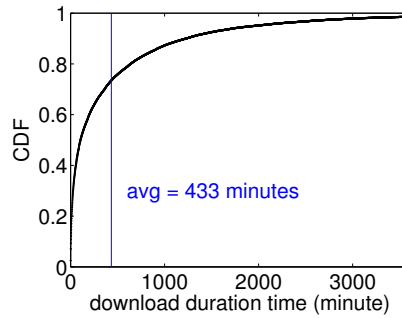


Figure 13: CDF of download duration time.

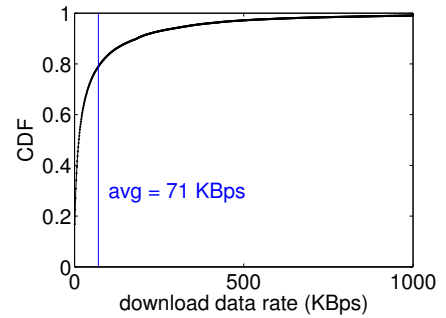


Figure 14: CDF of download data rate.

The average data transfer rate is 202 KBps ( $> 1.6$  Mbps). As shown in Figure 12, over 80% of the data transfer rates are more than 37.5 KBps (= 300 Kbps) which is the basic playback rate of online videos [2, 21]. The main reason why around 20% of the data transfer rates are less than 300 Kbps is that the user does not locate at the four major ISPs in China (as mentioned in Section 3.2) and thus his video retrieve data flow crosses multiple ISPs. More in detail, the data transfer rate in each ISP is plotted in Figure 15.

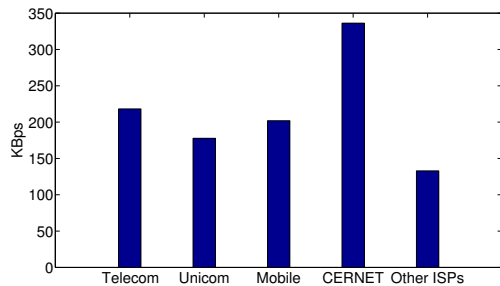


Figure 15: Data transfer rate in each ISP.

We discover a simple but accurate relation among the average file size, the average data transfer rate, and the average retrieve duration time, that is

$$\frac{\text{the avg file size (379 MB)}}{\text{the avg data transfer rate (202 KBps)}} = \text{the avg retrieve duration time (32 minutes)}. \quad (1)$$

This is a very useful equation which illustrates that if we take some measures (e.g., to invest more upload servers with higher upload bandwidth, or deploy more upload servers in more ISPs) to further enhance the data transfer rate, the retrieve duration time is expected to decrease inverse-proportionally. Thereby, we can find a proper tradeoff point between the data transfer rate and the retrieve duration time for future design.

#### 4.4 View Startup Delay

View startup delay is effectively alleviated by the implicit and secure data reuse among the users in VideoCloud. For a video request, if the cache hit status is 1-hit (i.e., the requested video has been stored in the cloud cache), the view startup delay is taken as 0; otherwise, the view startup delay is the download duration time. Thus, we firstly present the CDF of download duration time (in Figure 13) and the CDF of download data rate (in Figure 14), and then get the amortized view startup delay.

The average download duration time is 433 minutes and the average download data rate is 71 KBps (= 0.57 Mbps). In the previous subsection, we discover the relationship (see Equation (1)) among the average file size, the average data transfer rate, and the average retrieve duration time. However, this relationship does not exist as to the download duration time and the download data rate:

$$\frac{\text{the avg file size (379 MB)}}{\text{the avg download data rate (71 KBps)}} = 91 \text{ minutes} \ll \text{the avg download duration time (433 minutes)}. \quad (2)$$

The reason is that most download events have too low

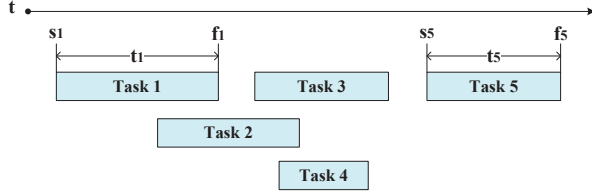


Figure 16: Download parallelism of a user  $i$ .

download data rate, e.g., 22% of the download rates are less than 2 KBps, 45% less than 10 KBps, and 70% less than 37.5 KBps (= 300 Kbps). In particular, 79% of the download data rates are less than the average download data rate (71 KBps).

As mentioned in Section 3.3, the data reuse rate among the users in VideoCloud reaches 87%, so the amortized view startup delay =  $87\% \cdot 0 + 13\% \cdot 433$  minutes = 56 minutes. We will make efforts to further reduce the amortized view startup delay, although this may be a challenging work.

## 4.5 Energy Efficiency

It is difficult for VideoCloud to record the accurate energy cost of each user, so we leverage the approximate power consumption of a PC (personal computer) to estimate the average energy cost of the users. According to a recent IDC report [38], laptop sales have occupied about 70% of the PC sales. And using the average statistics of the Dell PCs [39], the power consumption of a laptop is about 50 W and that of a desktop is about 250 W. Thereby, the average power consumption of a PC is estimated as

$$P_1 = 50W \cdot 70\% + 250W \cdot 30\% = 110W.$$

Our 17-day system log includes the performance information of  $N = 3.87M$  video requests. Without cloud download, the energy cost for the users to accomplish all the video requests is denoted as  $E_1$ . To properly estimate  $E_1$ , the *download parallelism* of each user must be taken into account. As shown in Figure 16, in the common download way, suppose the user  $i$  starts Task 1 at the time  $s_1$  and finishes Task 1 at the time  $f_1$ , and then the download duration time of Task 1 is  $t_1 = f_1 - s_1$ . The meanings of the symbols for Task 2/3/4/5 are similar. Then the *download parallelism* of the user  $i$  is computed as

$$p_i = \left( \sum_{j=1}^5 t_j \right) / (f_3 - s_1 + f_5 - s_5),$$

and the energy cost of the user  $i$  is computed as

$$e_i = P_1 \cdot (f_3 - s_1 + f_5 - s_5) = P_1 \cdot \left( \sum_{j=1}^5 t_j \right) / p_i.$$

As a result,  $E_1$  is estimated as

$$E_1 = \sum_{i=1}^m e_i = 2.21M \text{ KWH},$$

where  $m$  denotes the total number of users and KWH = Kilowatt Hour.

By using cloud download, the energy cost of the whole system is:

$$E_2 = E_c + E_u,$$

where  $E_c$  denotes the energy cost of the cloud utilities and  $E_u$  denotes the energy cost of the users. Since each server in the cloud works in all the time and the average power consumption of a server is  $P_2 = 700W$ ,  $E_c$  is estimated as:

$$E_c = P_2 \cdot S \cdot 17 \text{ days} = 700W \cdot 426 \cdot 17 \text{ days} = 0.12M \text{ KWH},$$

where  $S$  is the total number of servers in VideoCloud. And  $E_u$  is estimated in the same way we estimate  $E_1$ :

$$E_u = \sum_{i=1}^m e'_i = 0.18M \text{ KWH},$$

where  $e'_i$  is the energy cost of the user  $i$  in retrieving his requested videos from the cloud. Therefore,

$$E_2 = E_c + E_u = 0.12M \text{ KWH} + 0.18M \text{ KWH} = 0.3M \text{ KWH}.$$

In conclusion, the *user-side* energy efficiency is

$$\frac{E_1 - E_u}{E_1} = \frac{2.21M - 0.18M}{2.21M} = 92\%, \quad (3)$$

and the *overall* energy efficiency of the total cloud download system (cloud + users) is

$$\frac{E_1 - E_2}{E_1} = \frac{2.21M - 0.3M}{2.21M} = 86\%. \quad (4)$$

## 5. CONCLUSION AND FUTURE WORK

Video content distribution is becoming a kill application of the Internet owing to the users' inflating requirements on both video quantity and quality, so it is of great importance to investigate how to achieve high-quality content distribution for both popular and unpopular videos. In this paper, we firstly analyze the state-of-the-art techniques. We find they generally work well in distributing popular videos, but do not provide satisfactory content distribution service for unpopular videos. In fact, there are many more unpopular videos than popular videos over the Internet. Therefore, we propose and implement the novel cloud download scheme, which utilizes cloud utilities to achieve high-quality content distribution for unpopular videos. Running logs of our real deployed commercial system confirm the effectiveness and efficiency of cloud download. Our study provides practical experiences and valuable heuristics for making use of cloud utilities to achieve efficient Internet services.

Still some future work remains. Firstly, it is possible to extend the application of cloud download to smaller or private organizations. In our opinion, even a company or a university can build a cloud download system like VideoCloud as its private cloud system to facilitate its users. For example, we discover that in many companies some employees may keep their computers all night on to continue an unfinished download task after they leave the company, which is rather *energy-inefficient*. Since the design of cloud download (described in Section 3) is quite simple and practical, a company can invest in building a private (small-scale) cloud download system and then encourage its employees to send every download request to the cloud.

Secondly, as mentioned in Section 3.3, although cloud download has improved the download success rate of requested videos to 81.4%, download failure still exists. In fact, it is very difficult to judge the download failure of a video request. Given that the most powerful web search engine has just discovered less than 1% of all the things existing in the Internet [40], it is impossible to tell a user whether his

requested video can be obtained at last if we keep on trying. Consequently, we have to tell the user his video request is *judged to have failed* at the “right” time. Then the key point is: what is the “right” time? In another words, what is the rule to judge the failure of a video request? Presently, we choose a simple judging rule which takes into account both the download duration time and the download progress (i.e., the download completion fraction of the requested video). The rule is: check the download progress periodically; if the download progress does not change in the latest period, the video request is judged to have failed. The current period is empirically set to be 24 hours. Next step we plan to further analyze the system log to discover more factors that influence the failure of video requests and consider a more accurate failure judging algorithm.

Finally, as a large-scale commercial system in its first version, VideoCloud tends to adopt traditional/straightforward design in constructing each component so that the deployment and debugging are easy to handle. We realize there is still considerable optimization space for novel/original design to take effect. This paper is the first step of our efforts, and we are dedicated to obtaining more lessons and insights on our way.

## 6. ACKNOWLEDGEMENTS

This work is supported by the National Basic Research Program of China under Grant 2011CB302305 and the National Natural Science Foundation of China under Grants 61073015 and 60873051. We would like to thank Fuchen Wang and Yue Cao for their valuable help.

## 7. REFERENCES

- [1] The Cisco Visual Networking Index report. [http://newsroom.cisco.com/dlls/2008/ekits/Cisco\\_Visual\\_Networking\\_Index\\_061608.pdf](http://newsroom.cisco.com/dlls/2008/ekits/Cisco_Visual_Networking_Index_061608.pdf).
- [2] YouTube we site. <http://www.youtube.com>.
- [3] The BitTorrent protocol specification. [http://www.BitTorrent.org/beps/bep\\_0003.html](http://www.BitTorrent.org/beps/bep_0003.html).
- [4] PPLive web site. <http://www.pptv.com>.
- [5] X. Zhang, J. Liu, B. Li, and T. Yum. “CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming,” In IEEE INFOCOM, 2005.
- [6] C. Huang, J. Li, and K. Ross. “Can Internet Video-on-Demand be Profitable?” In ACM SIGCOMM, 2007.
- [7] Akamai web site. <http://www.akamai.com>.
- [8] J. Kangasharju, J. Roberts, and K.W. Ross. “Object replication strategies in content distribution networks,” Computer Communications, vol. 25, no. 4, 2002, pp. 376 - 383.
- [9] eMule web site. <http://www.eMule.org>.
- [10] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and others. “Above the Clouds: A Berkeley View of Cloud Computing,” Berkeley Tech. Rep., 2009.
- [11] A. Greenberg, J. Hamilton, D.A. Maltz, and P. Patel. “The cost of a cloud: research problems in data center networks,” SIGCOMM Computer Communication Review, vol. 39, no. 1, 2008, pp. 68 - 73.
- [12] K. Roth and K. McKenney. “Energy Consumption by Consumer Electronics in U.S. Residences,” Final Report to the Consumer Electronics Association (CEA), Jan. 2007.
- [13] J. R. Douceur. “The sybil attack,” In IPTPS 2002, Cambridge, MA, March 2002, pp. 251 - 260.
- [14] A. Singh, T. Ngan, P. Druschel, D. Wallach. “Eclipse attacks on overlay networks: Threats and defenses,” In IEEE INFOCOM, 2006.
- [15] E. Sil and R. Morris. “Security considerations for peer-to-peer distributed hash tables,” In IPTPS, 2002.
- [16] The VideoCloud web page (March 2011). [http://xf.qq.com/help\\_video.html](http://xf.qq.com/help_video.html).
- [17] The QQCyclone platform. <http://xf.qq.com>.
- [18] Amazon S3 web site. <http://aws.amazon.com/s3>.
- [19] MS Azure. <http://www.microsoft.com/windowsazure>.
- [20] Dropbox web site. <http://www.dropbox.com>.
- [21] Youku web site. <http://www.youku.com>.
- [22] D. Wu, Y. Hou, W. Zhu, Y. Zhang, and J. Peha. “Streaming video over the Internet: approaches and directions,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 3, 2001.
- [23] The ChinaCache CDN web site. <http://www.chinacache.com>.
- [24] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. “Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky,” In ACM Multimedia, 2009, pp. 25 - 34.
- [25] C. Wu, B. Li, and S. Zhao. “On Dynamic Server Provisioning in Multichannel P2P Live Streaming,” to appear in IEEE/ACM Transactions on Networking.
- [26] UUSee web site. <http://www.uusee.com>.
- [27] <http://www.webtvwire.com/category/internet-hdtv>.
- [28] F. Liu, S. Shen, B. Li, B. Li, H. Yin, and S. Li. “Novasky: Cinematic-Quality VoD in a P2P Storage Cloud,” In IEEE INFOCOM, 2011.
- [29] China Telecom. <http://www.chinatelecom.com.cn>.
- [30] China Unicom. <http://www.chinaunicom.com.cn>.
- [31] China Mobile. <http://www.10086.cn>.
- [32] CERNET (China Education and Research Network) web site. <http://www.cernet.edu.cn>.
- [33] H. Xie, Y.R. Yang, A. Krishnamurthy, Y.G. Liu, and A. Silberschatz. “P4P: Provider portal for applications,” In ACM SIGCOMM, 2008.
- [34] D.R. Choffnes, and F.E. Bustamante. “Taming the torrent,” In ACM SIGCOMM, 2008.
- [35] Amazon Cloud Drive web site. <https://www.amazon.com/cloudrive/learnmore>.
- [36] S. Podlipnig and L. Boszormenyi. “A survey of web cache replacement strategies,” ACM Computing Surveys, vol. 35, no. 4, 2003, pp. 374 - 398.
- [37] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. “Web caching and Zipf-like distributions: Evidence and implications,” In IEEE INFOCOM, 1999, pp. 126 - 134.
- [38] The IDC Desktop and Laptop Sales report. <http://news.techworld.com/sme/3227696/desktop-and-laptop-sales-to-grow-this-year>.
- [39] Dell web site (March 2011). <http://www.dell.com>.
- [40] B. Croft, D. Metzler, and T. Strohman. “Search engines: Information retrieval in practice,” Addison-Wesley, USA, 2009.