Li ZH, Liu G, Ji ZY *et al.* Towards cost-effective cloud downloading with Tencent big data. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(6): 1163–1174 Nov. 2015. DOI 10.1007/s11390-015-1591-5

Towards Cost-Effective Cloud Downloading with Tencent Big Data

Zhen-Hua Li[1,2] (李振华), *Member, CCF, ACM, IEEE*, Gang Liu[3] (刘 刚), Zhi-Yuan Ji[4,*] (嵇智源) and Roger Zimmermann[5], *Senior Member, IEEE, Member, ACM*

[1]*School of Software, Tsinghua University, Beijing 100084, China*

[2]*Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China*

[3]*QQXuanfeng System Group, Tencent Co., Ltd, Shanghai 200233, China*

[4]*High Technology Research and Development Center, Ministry of Science and Technology, Beijing 100044, China*

[5]*Department of Computer Science, National University of Singapore, Singapore 117417, Singapore*

E-mail: lizhenhua1983@tsinghua.edu.cn; ganghust@qq.com; jzy@htrdc.com; rogerz@comp.nus.edu.sg

Received March 12, 2015; revised August 4, 2015.

Abstract The cloud downloading scheme, first proposed by us in 2011, has effectively optimized hundreds of millions of users' downloading experiences. Also, people start to build a variety of useful Internet services on top of cloud downloading. In brief, by using cloud facilities to download (and cache) the requested file from the "best-effort" Internet on behalf of the user, cloud downloading ensures the data availability and remarkably enhances the data delivery speed. Although this scheme seems simple and straightforward, designing a real-world cloud downloading system involves complicated and subtle trade-offs (between deployment cost and user experience) when serving a large number of users: 1) how to plan the cloud cache capacity to achieve a high and affordable cache hit ratio, 2) how to accelerate the data delivery from the cloud to numerous users, 3) how to handle the dense user requests for highly popular files, and 4) how to judge a potential downloading failure of the cloud. This paper addresses these design trade-offs from a practical perspective, based on big data from a nationwide commercial cloud downloading system, i.e., Tencent QQXuanfeng. Its running traces help us find reasonable design strategies and parameters, and its real performances confirm the efficacy of our design. Our study provides solid experiences and valuable heuristics for the developers of similar and relevant systems.

Keywords cloud downloading, cost-effective, design trade-off, big data

1 Introduction

Downloading is a basic service the Internet offers to its users. Nevertheless, "high-quality" downloading service is still desired in today's "best-effort" Internet environments. Here the meaning of high-quality is two-fold: 1) ensured data availability, and 2) high data delivery speed. Ensured data availability implies that at least one full copy of the requested file is available for the user(s). Meanwhile, the significance of high data delivery speed lies in three aspects: 1) reducing the download time, 2) saving the battery consumption of mobile devices[1], and 3) enabling the user-friendly video streaming service as the majority of downloading traffic is ascribed to videos.

Existing downloading techniques mainly include the C/S (client/server), CDN (content delivery network, such as Akamai, L-3, and ChinaCache), and P2P (peer-to-peer) approaches. The C/S approach is obviously subject to the single-point bottleneck of the centralized server (cluster), and thus lacks scalability. The CDN approach optimizes downloading performance by deploying edge servers in numerous locations that are closer to users. However, as a commercial service, CDN vendors typically only help to deliver (relatively popular) files for those content providers who pay for the

Regular Paper

Special Section on Networking and Distributed Computing for Big Data

This work is sponsored by the National Natural Science Foundation of China under Grant Nos. 61471217 and 61472266, the China Postdoctoral Science Fund under Grant No. 2014M550735, and the CCF-Tencent Open Fund under Grant No. AGR20150201.

*Corresponding Author

service. On the other hand, the P2P approach mainly relies on the often unstable but enormous end systems owned by users to form data swarms[2-3]. The real strength of P2P lies in its efficacy in distributing popular files that are each shared by a number of peers. With regard to an unpopular file, for the lack of corresponding swarms/peers, it is generally hard to guarantee the data availability or to maintain a high downloading speed. In a nutshell, all the existing techniques are unsatisfactory under certain scenarios.

The emergence of cloud services[4] offers a new opportunity to address the challenging problem of efficiently downloading various files from various data sources. The cloud downloading scheme, first proposed by us in 2011[5], has effectively optimized hundreds of millions of users' downloading experiences up to now[6-7]. The major commercial cloud downloading systems include Tencent QQXuanfeng①, Xunlei②, Baidu CloudDisk③, and so forth.

The basic working principle of cloud downloading is depicted in Fig.1. First, a user sends his/her downloading request to the cloud. Subsequently, the cloud downloads the requested file from the Internet on behalf of the user, and stores a copy in the cloud cache. Then, the user can retrieve his/her requested file from the cloud with a high speed. In a word, by making use of cloud facilities, cloud downloading ensures the data availability and remarkably enhances the data delivery speed. Note that the business model of a cloud downloading system is the opposite of a CDN, because CDNs only serve paid content providers, whereas a cloud downloading system charges its users (i.e., content receivers) for better downloading experiences.

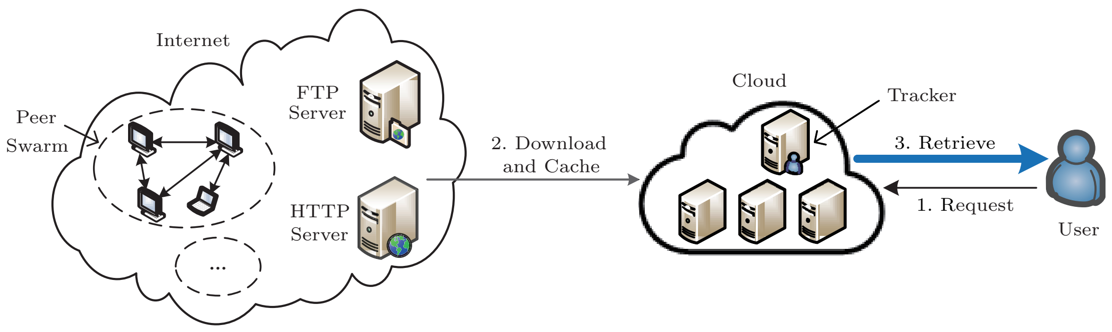


Fig.1. Basic working principle of cloud downloading.

Although the cloud downloading scheme seems simple and straightforward, designing a real-world cloud downloading system involves complicated and subtle trade-offs (between deployment cost and user experience) when it serves a large number of users. According to our practical experiences in deploying, maintaining, and optimizing a nationwide cloud downloading system, i.e., Tencent QQXuanfeng, the major design trade-offs need to address the following four questions.

• *How to plan the cloud cache capacity to achieve a high and affordable cache hit ratio?* Ideally, the cloud should permanently cache every downloaded file to ensure its data availability forever, and then the cache hit ratio would be the highest. But in practice, the required storage cost for permanently caching everything is unaffordable. By measuring the retrieval delay of all requested files, we find that almost all requested files are retrieved within 12 days since they were requested. Thus, QQXuanfeng only caches a newly downloaded file for 12 days to achieve a high and affordable cache hit ratio. Besides, we study the cloud cache replacement strategy in Subsection 3.1, in particular the LFU strategy and its "frequency aging effect".

• *How to accelerate the data retrieval from the cloud to numerous users?* Given that cross-ISP data delivery performance seriously degrades and inter-ISP network traffic cost is expensive in China, we address this problem via the intra-cloud ISP-aware data uploading technique which will be elaborated in Subsection 3.2. In brief, we deploy specialized uploading servers in multiple major ISP networks and always attempt to restrict the data delivery flow to within the same ISP network

①http://xf.qq.com, Sept. 2015.
②http://lixian.vip.xunlei.com, Sept. 2015.
③http://pan.baidu.com, Sept. 2015.

as a user's. Though this technique looks quite simple, it is effective in most cases.

• *How to handle the dense user requests for highly popular files?* The key strength of cloud downloading lies in delivering unpopular files[5]. But as the user scale grows, a considerable portion of user requests are issued for highly popular files. These dense downloading requests for highly popular files result in a severe bandwidth burden (including "flash crowds") on the cloud. To this end, we develop a special "tracker" to monitor online users and their shared files. When a user wants to retrieve a file f, he/she is first directed by the Tracker to other online users (sharing f) for peer-assisted data delivery (which will be elaborated in Subsection 3.3). If the peer-assisted data delivery speed is unsatisfactory, QQXuanfeng would provide extra cloud bandwidth for further acceleration. Thereby, the severe cloud bandwidth burden for delivering highly popular files can be effectively alleviated.

• *How to judge a potential downloading failure of the cloud?* It is impossible to accurately judge whether a requested file can be eventually downloaded if we keep on trying. Instead, we have to notify the user that her/his downloading request is judged to have failed at the "right" time so that the user does not need to wait endlessly. Then the critical question is: what is(are) the rule(s) to select the "right" time? Based on empirical data, we employ a combination of three rules: 1) after QQXuanfeng receives a downloading request, if it cannot download any part of the requested file in five minutes, the downloading request is judged to have failed; 2) QQXuanfeng periodically examines the downloading progress of each requested file, and if the downloading progress stagnates (i.e., does not change) in a certain period, the corresponding downloading request is also judged to have failed; 3) if QQXuanfeng cannot finish downloading a requested file in a whole day, it recommends the user to give up. More details will be presented in Subsection 3.4.

By the end of 2013, the number of registered users of QQXuanfeng had exceeded 26 million, and the number of daily received downloading requests had reached 0.72 million. The whole system utilized 663 commodity servers, and the cloud cache accommodated 800 TB of unique data. Users were charged according to their allocated cloud storage space, regardless of their bandwidth/traffic consumed. The reason is straightforward: a user of QQXuanfeng usually retrieves his/her requested file from the cloud at most once, and thus his/her bandwidth/traffic consumed is basically dependent on his/her allocated cloud storage space.

Real performance data of QQXuanfeng confirm the efficacy of our design. The average data delivery speed (from the cloud to users) is as high as 2.1 Mbps, where 81% of delivery speeds exceed 300 Kbps — the basic playback rate of online videos. In comparison, the average data delivery speed of common downloading is merely 0.55 Mbps. Common downloading denotes the common way in which a user downloads a file from the Internet, i.e., by using a common web browser or a P2P client software. The cloud cache hit ratio reaches 87%, and the downloading failure ratio is as low as 7.8%. Further, when users retrieve highly popular files from QQXuanfeng, 78% of network traffic originates from peering users, thus greatly reducing the cloud bandwidth burden.

Nowadays, people start to build a variety of useful Internet services on top of cloud downloading, such as cloud-based media converters[8] and cloud-accelerated web browsers (e.g., QQ mobile web browser, UCWeb browser, and Amazon Silk web browser). For example, with the help of cloud downloading, a cloud-based media converter only requires its users to upload the media links rather than the original media files, to avoid unnecessary uploading traffic and energy consumption of the end user's equipment. This is especially helpful for those battery-operated small-screen mobile devices. Our study of cloud downloading provides solid experiences and valuable heuristics for the developers of similar and relevant systems.

2 System Overview

The system architecture of QQXuanfeng is made up of six components: 1) ISP proxies, 2) task manager, 3) task dispatcher, 4) downloaders, 5) cloud cache, and 6) tracker, as depicted in Fig.2. The detailed hardware configuration of each component is listed in Table 1. All the information of memory, storage, and bandwidth refers to one server. Overall, the system utilizes 663 commodity servers, including 400 chunk servers that constitute an 800 TB cloud cache, 140 download servers with nearly 45 Gbps of Internet bandwidth, 93 uploading servers with around 20 Gbps of Internet bandwidth, and so forth. Each server runs the SUSE Enterprise Linux (version 10.1) operating system. Below we overview the whole system by following the message/data flows with regard to a typical cloud downloading task.

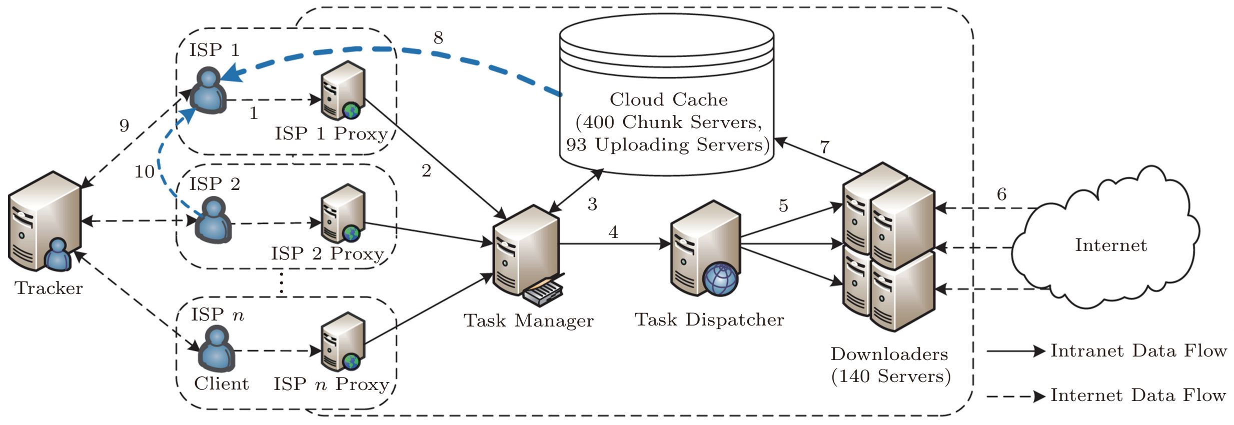


Fig.2. System architecture of QQXuanfeng (in Dec. 2013).

Table 1. Hardware Configuration of QQXuanfeng (in Dec. 2013)

Building Block	Number of Servers	Memory (GB)	Storage	Internet Bandwidth (Gbps)
ISP proxy	6	8	250 GB	0.300
Task manager	4	8	250 GB	—
Task dispatcher	3	8	460 GB	—
Downloaders	140	8	460 GB	0.325
Cloud cache	496 = 400 + 93 + 3	8	4 TB (chunk server)	0.300
Tracker	14	8	150 GB	0.300

The user of QQXuanfeng should have installed its client software which can be downloaded from the QQXuanfeng website[4]. The client software is able to recognize which ISP network the user belongs to from the user's IP address, thereby the downloading request is first sent to the corresponding ISP proxy (see arrow 1 in Fig.2). Each ISP proxy maintains a downloading request queue to restrict the number of downloading requests sent to the task manager (see arrow 2 in Fig.2), so that the task manager is resilient to possible request upsurges or DoS (denial-of-service) attacks in any ISP network. By the end of 2013, QQXuanfeng has deployed four ISP proxies in the four major ISP networks in China: Telecom, Unicom, Mobile, and CERNET. If the user does not belong to any of the abovementioned ISP networks, his/her downloading request is simply sent to a random ISP proxy.

When receiving a downloading request, the task manager first examines whether the requested file has a copy in the cloud cache (see arrow 3 in Fig.2). If the downloading request is a BitTorrent/eMule link, the task manager examines whether the cloud cache contains a file that has the same hash code with that contained in the BitTorrent/eMule link (since a BitTorrent/eMule link contains the hash code of its affiliated file in itself, while an HTTP/FTP link does not). Otherwise, the task manager directly examines whether the downloading link is repeated in the cloud cache. If the requested file actually has a copy, the user can then retrieve it from the cloud cache (see arrow 8 in Fig.2), from other online users (see arrow 10 in Fig.2), or from both sources in parallel. Otherwise, the task manager hands the request over to the task dispatcher (see arrow 4 in Fig.2).

The task dispatcher assigns the downloading request to one "downloader" server for data downloading from the Internet in a common way (see arrow 5 in Fig.2). For example, if the requested file is hosted in a P2P data swarm, the assigned downloader would act as a common peer to join in the data swarm. The task dispatcher is also responsible for balancing the bandwidth loads of the 140 downloaders. Here, the number of downloaders (140) is empirically determined to ensure that the total downloading bandwidth (45 Gbps) exceeds the peak load of downloading tasks ($\approx$ 40 Gbps), so that the view startup delay (which will be explained in Subsection 3.1) can be minimized. Each downloader executes multiple downloading tasks in parallel to make full use of its downloading bandwidth (see arrow 6 in Fig.2). Accordingly, the task dispatcher always assigns a newly

[4]http://xf.qq.com, Sept. 2015.

incoming downloading request to the downloader that has the lowest real-time downloading speed.

Once the downloader finishes a downloading task, it first computes the hash code of the downloaded file, and then attempts to store the file into the cloud cache (see arrow 7 in Fig.2). The downloader first examines whether the file has a copy in the cloud cache (using the hash code). If the file is duplicated, the downloader simply discards it. Otherwise, the downloader examines whether the cloud cache has enough spare space to accommodate the new file. If the cloud cache does not have enough spare space, it has to delete some cached files to obtain enough spare space for accommodating the new file. The cloud cache replacement strategy will be studied in Subsection 3.1.

The cloud cache is composed of 400 chunk servers, 93 uploading servers, and three index servers, as demonstrated in Fig.3. All these servers are connected by a DCN (data center network). The DCN adopts the traditional three-layer tree structure to organize its switches, consisting of a core layer in the root of the tree, an aggregation layer in the middle, and an edge layer at the leaves. Inside the cloud cache, every file is segmented into fixed-size chunks stored in the chunk servers, and every chunk has a duplicate for redundancy. As a result, the 400 chunk servers can accommodate a total of $\frac{400 \times 4\text{ TB}}{2} = 800$ TB of unique data.

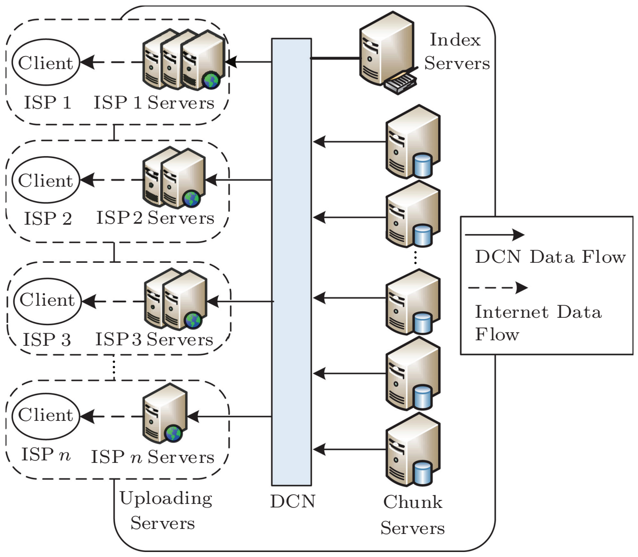


Fig.3. System architecture of the cloud cache.

To facilitate load balancing and exploit chunk correlations in the same file, all the chunks of a file are stored together in a chunk server that has the biggest real-time spare storage space. Meanwhile, duplicate chunks of this file must be stored in another chunk server. There is an index server as well as two backup index servers maintaining the metadata of chunks (for chunk search and validation). The metadata is organized as a list of n-tuples that look like (file hash code, file link, number of chunks, physical location of the first chunk, physical location of the first duplicate chunk, $\cdots$).

3 Design Trade-Off

This section addresses the major design trade-offs of cloud downloading in detail, based on the QQXuanfeng big data that helps us find reasonable design strategies and parameters[9].

3.1 Planning the Cloud Cache Capacity

The biggest hardware, network, and energy cost of the QQXuanfeng system originates from the cloud cache component. Cloud cache plays a critical role in the system by avoiding repeated downloading tasks, increasing the downloading success ratio, and reducing the view startup delay. If a user requests for a non-cached video file, he/she must first wait for the cloud to download it from the Internet. Thus, he/she cannot view the video immediately. Such waiting time is denoted as the view startup delay. Consequently, we want the cloud cache hit ratio to be as high as possible.

Ideally, if the cloud permanently caches every downloaded file, the cache hit ratio will be the highest, but the required storage space is obviously unaffordable in practice. By measuring the retrieval delay (if a file is requested by a user at time t_1 and is then retrieved by the user at time t_2, the corresponding retrieval delay is $t_2 - t_1$) of all the requested files in Dec. 2013, we find that almost all the requested files are retrieved within 12 days since they were requested. Specifically, the retrieval ratios corresponding to 1 day, 2 days, $\ldots$, 12 days are 70%, 11%, $\ldots$, 0.5%, respectively, while only 0.2% of the requested files are retrieved after 12 days. Thereby, to achieve a high and affordable cache hit ratio, QQXuanfeng only caches a newly downloaded file for 12 days. Note that once a cached file is requested again, its cache duration is reset to 12 days. If a file has been cached for longer than its cache duration, the user who issued the request will get notified by QQXuanfeng.

In Dec. 2013, the maximum number of daily requests served by QQXuanfeng was 0.85M (M = million), and QQXuanfeng is expected to handle up to

1.0M daily requests. Since the average size of requested files is 390 MB (refer to Fig.6) and a newly downloaded file is cached for 12 days, the total cloud cache capacity should be:

$$C_{\text{cache\_no\_hit}} \approx 390 \text{ MB} \times 1.0\text{M} \times 12 = 4\,680 \text{ TB},$$

to well handle 1.0M daily requests when the cloud cache hit ratio is 0. According to the running logs of QQXuanfeng, the cloud cache hit ratio generally lies in between 83% and 91% (87% on average). Taking the worst case (83%) for conservative calculation, we plan the cache capacity as:

$$C_{\text{cache}} \approx 390 \text{ MB} \times 1.0\text{M} \times 12 \times (1 - 83\%) = 796 \text{ TB}.$$

Further, to cope with the possible fluctuations, the total cloud cache capacity is finally planned as $C_{\text{cache}} =$ 800 TB, slightly larger than 796 TB.

Cache Replacement Strategy. As QQXuanfeng becomes more and more popular, it is quite possible that the daily number of download requests will be much higher than 1.0M some day. If the daily number of download requests increases to 10M, what shall we do? It is infeasible for us to construct a 8000-TB cloud cache made up of 4 000 chunk servers — the cost is far beyond our current affordability. As a result, some data must be replaced to provide cost-effective utilization of the cloud cache capacity, where the cache replacement strategy plays a key role.

We first investigate the performance of the three commonly used cache replacement strategies, i.e., FIFO (first in first out), LRU (least recently used), and LFU (least frequently used) via trace-driven simulations. The trace is a 14-day system log (refer to Subsection 4.1 for detailed information) of QQXuanfeng. As shown in Fig.4, among the three common strategies, FIFO performs the worst, and LFU performs the best. The simulated cache capacity is configured as 100 TB, and using larger cache capacities generates similar results.

Although LFU works best among the three common strategies, it is known that LFU bears a non-negligible "frequency aging" problem[10] (also called the "cache pollution" problem) that may well harm its long-term performance. Specifically, when LFU is applied, files that used to be highly popular during one time period tend to always remain in the cache, even when they have not been requested for a long time period. In other words, they "pollute" the cache. To address this problem, we must take into account the "aging" effect of the requesting frequencies of files. A typical solution is to utilize a periodic aging function[11-12]:

$$f_{i+1}(c) = \alpha \times f_i(c), \quad 0 < \alpha < 1,$$

where $f_i(c)$ denotes the aging frequency of file c in the i-th aging period (T), and α denotes the aging ratio. Typically, T is set to 24 hours and α is set to 0.5. The performance of the resulting "typical LFU-aging" strategy is also plotted in Fig.4. Obviously, the typical LFU-aging strategy is always better than FIFO and LRU, and it outperforms LFU from the 8th day on.

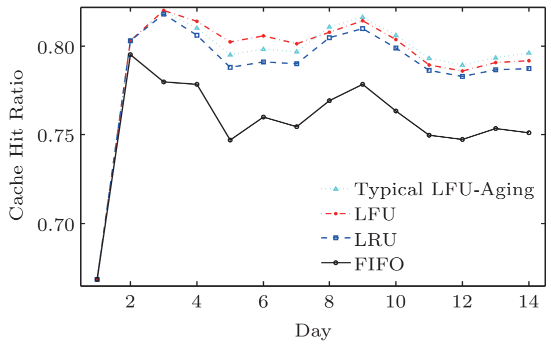


Fig.4. Cache hit ratios of the three common cache replacement strategies: FIFO, LRU, and LFU, as well as one "typical LFU-aging" strategy where the aging period (T) is 24 hours and the aging ratio (α) is 0.5.

3.2 Accelerating the Data Retrieval

For an unpopular file, a fundamental problem of cloud downloading is how to accelerate the data retrieval process, so that the user can retrieve it with a high speed. Considering that cross-ISP data delivery performance degrades seriously and inter-ISP network traffic is often expensive in China, we address this problem via the intra-cloud ISP-aware data uploading technique. In brief, we deploy specialized uploading servers in multiple ISP networks (see Fig.3) and always attempt to restrict the data flow to within the same ISP network as the user's.

However, as users' requests come from tens of ISP networks, in practice, we are unable to deploy uploading servers in every involved ISP network. Observing that the majority of downloading requests come from the four major ISP networks in China, we only deployed uploading servers in these four ISP networks. And the number of uploading servers deployed in each ISP network is mainly proportional to the corresponding network traffic volume. If a user belongs to other ISP networks, we choose an uploading server that has the lowest real-time uploading burden to serve him/her.

In detail, suppose a user locating at ISP 1 wants to retrieve a file f stored in a chunk server S, and QQXuanfeng has placed 10 uploading servers (U_1, U_2, $\cdots$, U_{10}) in ISP 1. The chunks of f are first delivered from S to a certain uploading server (say, U_4) in ISP 1, and then delivered from U_4 to the user. The delivery process is in a "pass-through" manner — as soon as U_4 gets a complete chunk of f, U_4 delivers the chunk to the user. f is not cached in U_4, because the intra-cloud (DCN) delivery bandwidth is high (1 Gbps) and we do not need to make things unnecessarily complicated.

3.3 Handling the Dense User Requests for Highly Popular Files

The real strength of cloud downloading lies in delivering unpopular files. However, as its user scale grows, we discover that a considerable portion (43%) of requests are issued for highly popular files. Here "highly popular" is empirically taken as: receiving more than 10 requests per day (in the latest two weeks). Inside the cloud cache, merely 0.45% of files are highly popular at present, which confirms the massive request density of highly popular files. These dense requests for highly popular files result in a severe bandwidth burden (including "flash crowds") on the cloud.

To this end, we develop a special "tracker" to monitor online users and their shared files. The tracker maintains a list of 3-tuples (see arrow 9 in Fig.2). Each 3-tuple looks like: (user ID; user IP address:port; hash code (of the shared file); $\cdots$). Whenever a user joins/leaves the QQXuanfeng system or changes his/her shared files, he/she must notify the tracker. Meanwhile, every online user sends a beacon message to the tracker per five minutes to report his/her latest working status.

When a user wants to retrieve a file f from QQXuanfeng, he/she first asks the tracker whether there are other online peers sharing f. If not, the user directly retrieves f from the cloud; otherwise, the user is directed (by the tracker) to its online peers sharing f for peer-assisted delivery. Note that the peer-assisted delivery mechanism is only experimented on a portion of PC users, rather than mobile users of QQXuanfeng. This is because mobile devices (like smartphones and tablets) consume energy more quickly with peer-assisted data transfer[1]. If the average peer-assisted delivery speed is satisfactory ($>$ 2 Mbps), the user only retrieves data chunks of f from his/her peers; otherwise, the user asks QQXuanfeng for extra cloud bandwidth, trying to further accelerate the user's total retrieval speed to more than 2 Mbps.

A user can maintain at most 25 simultaneous TCP connections with his/her neighboring peers. Maintaining too many simultaneous TCP connections cannot enhance the data delivery speed, but brings about excessive control overhead. At the same time, the data delivery speeds of all the connections are examined every minute, in order to replace the worst connection(s) with one or several random new connections.

A file is segmented into chunks of equal size, and the chunk size should be a power of 2 bytes — typically between 32 KB and 16 MB. Every user in the peer swarm maintains a "chunk map" and sends the chunk map to his/her connected peers every minute if the chunk map is updated. The user's client (software) adopts a simple chunk scheduling strategy: at any time, it only assigns one chunk for one TCP connection to retrieve. When the assigned chunk is obtained, the corresponding TCP connection will be assigned to retrieve another chunk. The chunk retrieval priority depends on the user's requirement: if the user chooses the "view-as-download" mode, the chunks are retrieved in their playback order; otherwise, the chunks are retrieved in the "rarest-first" manner.

3.4 Judging a Potential Downloading Failure of the Cloud

Due to the high dynamics and heterogeneity of today's "best-effort" Internet, it is impossible to accurately judge whether a requested file can be eventually obtained if we keep on trying. Instead, we have to tell the user that his/her request is judged to have failed at the "right" time, so that the user does not need to wait endlessly. Then the key question is: what is(are) the rule(s) to select the "right" time? Based on the data analysis of QQXuanfeng, we employ a combination of three rules as follows.

• After receiving a downloading request, if QQXuanfeng cannot download any part of the requested file in five minutes, the downloading request is judged to have failed. This rule is confirmed by our sampling measurements indicating that if QQXuanfeng cannot download any part of the requested file in five minutes, QQXuanfeng is very likely (the probability is 97.3%) to fail in downloading the entire file in the subsequent whole day. In other words, the "startup status" of a downloading task is a good indicator of a potential downloading failure.

• QQXuanfeng periodically examines the downloading progress of each requested file. If the downloading progress stagnates (i.e., does not change) in the latest period, the downloading request is also judged to have failed. The period is empirically set as one hour, because one hour is considered to be neither too long (thus lagging in response) nor too short (thus being sensitive to network dynamics).

• If QQXuanfeng cannot download a requested file in a whole day, it recommends the user to give up. According to our records, 70% of the requested files are retrieved in one day, illustrating that most users expect QQXuanfeng to accomplish their downloading requests in one day. As a result, we do not want users to wait for longer than a whole day unless they accept this.

At last, Fig.5 plots the downloading success ratio of QQXuanfeng in 14 days. The ratio always stays above 90% and is 92.2% on average.

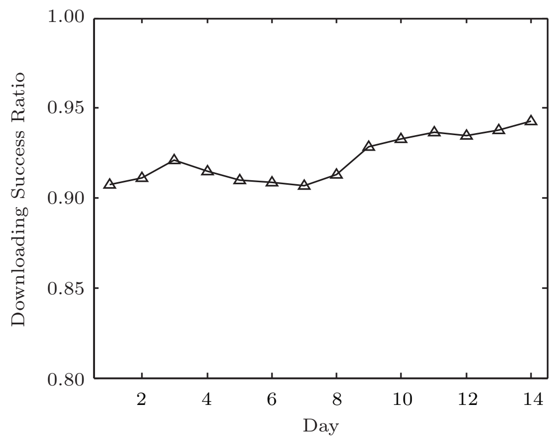


Fig.5. Downloading success ratio of QQXuanfeng.

4 Real-World Performance

4.1 Dataset

We use the running log of the QQXuanfeng system in 14 days in Dec. 2013 to evaluate the real-world performance of our design. The log includes the performance information of nearly 10 million downloading requests, involving 1.38 million unique files. 13.4% of the requested files are HTTP/FTP files, and the remaining are P2P (BitTorrent/eMule) files. For each downloading request, we record its user ID, file link, file hash code, file type, file size, issue time, downloading duration time (of the downloader), retrieval duration time (of the user), cache hit status (1-hit, 0-miss), cloud-based traffic, peer-assisted traffic, and so forth.

4.2 Metrics

• *Data retrieval speed* denotes the total data transfer rate when a user retrieves his/her requested file from the cloud and/or peering users. Specifically, the data retrieval speed is calculated as:

$$\frac{\text{cloud-based traffic} + \text{peer-assisted traffic}}{\text{retrieval duration time}}.$$

• *Peer-assisted traffic portion* denotes the portion of peer-assisted traffic in the data retrieval process. This portion represents to what extent the cloud bandwidth burden is alleviated by utilizing the peer-assisted file distribution.

• *View startup delay* denotes how long a user must wait for the cloud to download his/her requested file. For a downloading request, if the cloud cache hit status is "1-hit", the view startup delay is taken as 0; otherwise, the view startup delay is regarded as the corresponding downloading duration time of QQXuanfeng.

4.3 Data Retrieval Speed

Since the data retrieval speed is computed by dividing the file size (≈ the cloud-based traffic + the peer-assisted traffic) by the retrieval duration time, in this subsection, we first present the CDF of file size (in Fig.6), the CDF of retrieval duration time (in Fig.7), and the CDF of data retrieval speed (in Fig.8), and then explore their relations.

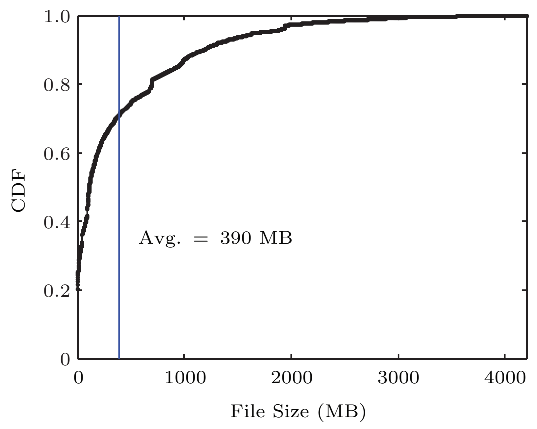


Fig.6. CDF of file size.

From Fig.6, we find that the average file size is 390 MB, and 25% of files are very small in size (< 8 MB). Fig.7 illustrates that the average retrieval duration time is 33 minutes. Besides, 76% of files are retrieved in less than 30 minutes, and 93% of files are retrieved in less than 100 minutes. As shown in Fig.8, the average data retrieval speed is 260 KBps ($\approx$ 2.1 Mbps). 81% of data retrieval speeds are higher than 37.5 KBps (= 300 Kbps) — the basic playback rate of online videos. Nevertheless, still 19% of data retrieval speeds are less than 300 Kbps. This is because the users are not all located in the four major ISP networks in China, and thus their data retrieval flows cross multiple ISP networks and are retarded by the "ISP barriers" (i.e., poor inter-connectivity between different ISP networks).

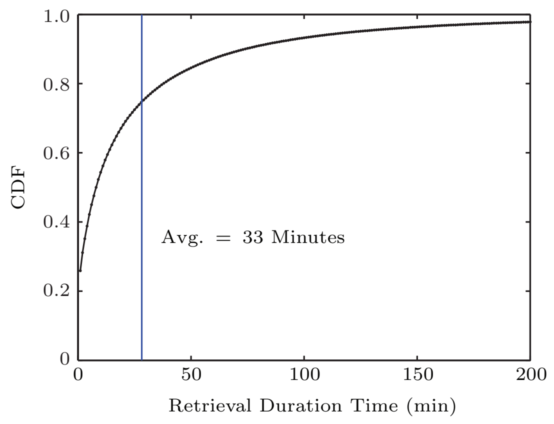


Fig.7. CDF of retrieval duration time.

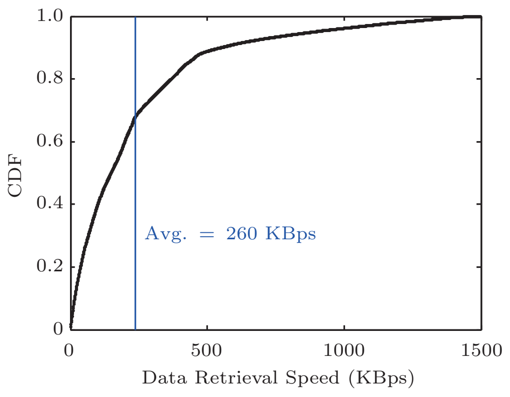


Fig.8. CDF of data retrieval speed.

Moreover, we discover an approximate relation among the average file size, the average data retrieval speed, and the average retrieval duration time:

$$\frac{\text{avg. file size (390 MB)}}{\text{avg. data retrieval speed (260 KBps)}} \approx \text{avg. retrieval duration time (33 min)}. \quad (1)$$

This is a useful equation implying that if we take certain measures (e.g., investing in more uploading servers or deploying more uploading servers in more ISP networks) to further enhance the data retrieval speed, the retrieval duration time is expected to decrease inverse-proportionally. Thereby, we can find a proper trade-off point between the data retrieval speed and the retrieval duration time for future design.

4.4 Peer-Assisted Traffic Portion

When a user wants to retrieve a (popular) file, he/she is first directed (by the tracker) to other online users sharing the file for peer-assisted delivery. For an unpopular file, the user can hardly find other online users sharing the file, thereby we do not (need to) consider unpopular files in this subsection. If the peer-assisted delivery speed is unsatisfactory, QQXuanfeng provides extra cloud bandwidth for further acceleration. Fig.9 records the peer-assisted traffic portion corresponding to various numbers of online peers sharing the file.

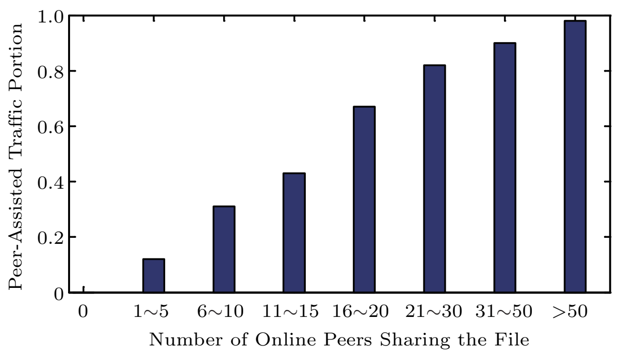


Fig.9. Peer-assisted traffic portion.

As we expected, more online peers usually bring about a larger peer-assisted traffic portion. When there are over 50 online peers sharing the same file, almost all the data (98%) is retrieved from peers, so that a "flash crowd" caused by a highly popular file will not happen to the system. In total, when users retrieve popular files from QQXuanfeng, 78% of the network traffic comes from peering users, thus effectively alleviating the cloud bandwidth burden.

4.5 View Startup Delay

The view startup delay is effectively reduced owing to the cloud cache. For a downloading request, if the requested file is already in the cloud cache, the view startup delay is just 0; otherwise, the view startup delay is the corresponding downloading duration time of

QQXuanfeng. Thus, we first present the CDF of downloading duration time (in Fig.10) and the CDF of downloading data rate (in Fig.11), and then calculate the amortized view startup delay.

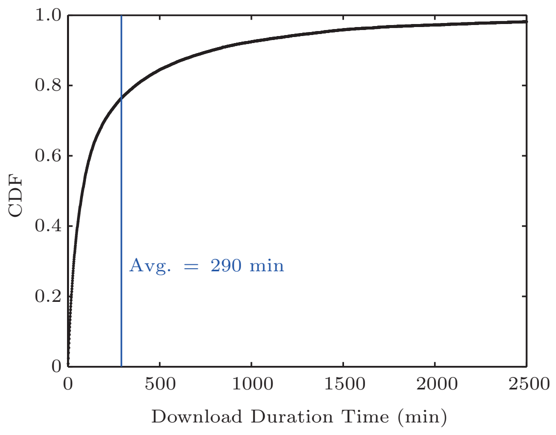


Fig.10. CDF of downloading duration time.

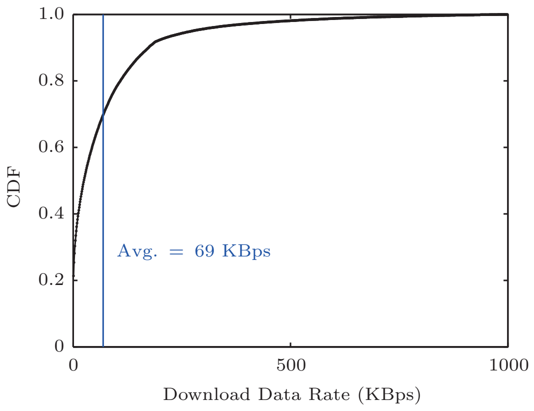


Fig.11. CDF of downloading data rate.

The average downloading duration time is 290 minutes, and the average downloading data rate is 69 KBps (= 0.55 Mbps). In Subsection 4.3, we discovered the relationship (see (1)) among the average file size, the average data retrieval speed, and the average retrieval duration time. But this relationship does not hold regarding the downloading duration time and the downloading data rate:

$$\begin{aligned}&\frac{\text{avg. file size (390 MB)}}{\text{avg. downloading data rate (69 KBps)}}\\ &= 96 \text{ min}\\ &\ll \text{avg. downloading duration time (290 min).}\end{aligned}$$

The reason is that most downloads have low data rates in the common downloading way — note that every downloader in the QQXuanfeng cloud downloads files in the common way. In particular, 38% of downloading rates are less than 10 KBps, 58% less than 37.5 KBps (= 300 Kbps), and 70% less than the average downloading data rate (69 KBps). As mentioned in Subsection 3.1, the cloud cache hit ratio in QQXuanfeng is 87% on average, thereby the amortized view startup delay $= 87\% \times 0 + 13\% \times$ 290 min = 37.7 min.

5 Related Work

In the past 15 years, large-scale high-quality content distribution has been a hot research topic in both industry and academia[13-15], especially in the state-of-the-art techniques: CDN and P2P. Many researchers have recognized that both CDN and P2P have their limitations. Thus, they have proposed various novel schemes by combining or optimizing CDN and P2P.

Hybrid CDN-P2P. On top of ChinaCache, the biggest commercial CDN service provider in China, Yin *et al.* developed a hybrid CDN-P2P live streaming system named LiveSky[16], to incorporate the strengths of both approaches: scalability of P2P and quality control capability of CDN. Although hybrid CDN-P2P inevitably results in extra deployment complexity and maintenance cost, LiveSky achieves a lower bandwidth burden on the CDN network and a higher working efficiency of peer streaming for very popular videos (i.e., live TV shows). However, for unpopular videos, it is still difficult and inefficient for few users to form a peer swarm, and then the performance of LiveSky will be similar to a common CDN.

P2SP (*Server-Assisted P2P*). Wu *et al.*[17] refocused on the importance of servers in P2P streaming and thus proposed the server-assisted P2P (or says "P2SP") streaming scheme. With a 7-month running trace of a large-scale commercial P2P live TV streaming system named UUSee[5], they found that the total available bandwidth of the 150 dedicated streaming servers could not meet the increasing demands of downloading bandwidth from hundreds of TV channels (a TV channel can be seen as a very popular peer swarm), although the total uploading bandwidth of peers also increased with downloading demands. Thereby, they proposed an allocation algorithm of server bandwidth named Ration, which proactively predicts the minimum server bandwidth demand of each channel and thus each

[5]http://www.uusee.com, Sept. 2015.

channel can be guaranteed with proper streaming quality. Ration has the similar shortcoming as LiveSky, because every TV channel of UUSee can be seen as a highly popular video. For unpopular videos, Ration will work like the traditional client/server (C/S) scheme.

Open-P2SP and Cloud-P2P. A more open, powerful, and complicated P2SP content distribution scheme (named "Open-P2SP") is presented in [18]. It integrates various third-party servers, contents, and data transfer protocols all over the Internet into a huge-scale and federated P2SP platform. As a result, Open-P2SP can overcome the limitations of traditional P2SP in both content abundance and server bandwidth. Further, if we regard all the third-party servers in an Open-P2SP system as a "cloud of clouds", Open-P2SP can be seen as a hybrid Cloud-P2P content distribution scheme. The key strength of Cloud-P2P lies in the so-called bandwidth multiplier effect. In order to maximize the overall bandwidth multiplier effect of a Cloud-P2P system, Li *et al.*[19] constructed a fine-grained performance model and developed a fast-convergent iterative algorithm named FIFA.

Smart AP-Assisted Cloud Downloading. In the recent two years, smart WiFi AP (access point, also known as home router) devices have quickly gained enormous popularity in China, e.g., HiWiFi[6], MiWiFi[7], and Newifi[8]. A traditional AP only forwards data packets for its connected end devices, while a smart AP can also pre-download and cache files on an embedded/connected storage device (e.g., an SD card, a USB flash drive, or a hard disk drive). In other words, smart APs can be used by people for "offline downloading". Li *et al.*[20] made a comparative study of cloud downloading and the smart AP-based offline downloading. They found that the cloud and the smart APs are complementary, while also being subject to distinct performance bottlenecks. Thereby, they designed a smart AP-assisted cloud downloading middleware called ODR (Offline Downloading Redirector) to help users achieve the best expected performance.

Xunlei Cloud Download System. To our knowledge, Xunlei is the biggest cloud download system of China at present. Unfortunately, until now there have been very few formal publications on its technical design or performance measurement, thus we can only obtain information from its web site and news media reports. Similar to QQXuanfeng, Xunlei also offers six levels (VIP1∼VIP6) of QoS for its users. QQXuanfeng generally allocates cloud storage space to its users in unit of 5 GB, because 5 GB is often the approximate size of a common TV play series, which we guess is also the reason for Xunlei's allocation scheme. The differences between Xunlei and QQXuanfeng are mainly three folds.

• Xunlei allocates "1 PB" of cloud storage space to its VIP4∼VIP6 users. "1 PB" seems astonishing and incredible, since a common Internet user is unlikely to have 1 PB of local storage. In fact, "1 PB" just means that Xunlei has built a large cloud cache, and the cache hit ratio is extremely high. As a result, Xunlei is confident in satisfying almost all the download requests of its VIP4∼VIP6 users.

• Xunlei sets the "maximum retrieval bandwidth" (i.e., the maximum data retrieval speed) for its users. On the contrary, QQXuanfeng does not set such limitation, and it always tries to accelerate its users' retrieval bandwidth as high as possible.

• The monthly charge of Xunlei increases linearly with the user level. Differently, QQXuanfeng charges its users at all levels the same (except for the "Trial" users). In order to get their level upgraded, the users of QQXuanfeng need to be more active (e.g., stay longer online) and more contributive (e.g., share more data with others).

6 Conclusions

In this paper, we reflected on the implementation techniques of the cloud downloading scheme, based on big data from a large-scale commercial cloud downloading system named QQXuanfeng. Although the scheme seems simple and straightforward, its real-world design involves complicated and subtle trade-offs between deployment cost and user experience. We addressed these design trade-offs from a practical, cost-effective perspective. Real performances of QQXuanfeng validate the efficacy and efficiency of our design.

References

[1] Liu Y, Guo L, Li F, Chen S. An empirical evaluation of battery power consumption for streaming data transmission to mobile devices. In *Proc. the 19th ACM-MM*, Nov.28-Dec.1, 2011, pp.473-482.

[2] Chen G, Li Z. Peer-to-Peer Network: Structure, Application and Design. Tsinghua University Press, Sep. 2007. (in Chinese)

[6] http://www.hiwifi.com, Sept. 2015.

[7] http://www.miwifi.com, Sept. 2015.

[8] http://www.newifi.com, Sept. 2015.

[3] Liu Y, Xiao L, Liu X, Ni L M, Zhang X. Location awareness in unstructured peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems*, 2005, 16(2): 163-174.

[4] Wen Y, Zhu X, Rodrigues J, Chen C. Cloud mobile media: Reflections and outlook. *IEEE Transactions on Multimedia*, 2014, 16(4): 885-902.

[5] Huang Y, Li Z, Liu G, Dai Y. Cloud download: Using cloud utilities to achieve high-quality content distribution for unpopular videos. In *Proc. the 19th ACM-MM*, Nov.28-Dec.1, 2011, pp.213-222.

[6] Ao N, Xu Y, Chen C, Guo Y. Offline downloading: A non-traditional cloud-accelerated and peer-assisted file distribution service. In *Proc. the 4th CyberC*, Oct. 2012, pp.81-88.

[7] Zhou Y, Fu Z, Chiu D M, Huang Y. An adaptive cloud downloading service. *IEEE Transactions on Multimedia*, 2013, 15(4): 802-810.

[8] Li Z, Huang Y, Liu G, Wang F, Zhang Z L, Dai Y. Cloud Transcoder: Bridging the format and resolution gap between Internet videos and mobile devices. In *Proc. the 22nd ACM NOSSDAV*, Jun. 2012, pp.33-38.

[9] Hu H, Wen Y, Chua T, Li X. Towards scalable systems for big data analytics: A technology tutorial. *IEEE Access*, 2014, 2: 652-687.

[10] Podlipnig S, Böszörmenyi L. A survey of web cache replacement strategies. *ACM Computing Surveys*, 2003, 35(4): 374-398.

[11] Zhang J, Izmailov R, Reininger D, Ott M. Web caching framework: Analytical models and beyond. In *Proc. IEEE Workshop on Internet Applications* (*WIA*), Jul. 1999, pp.132-141.

[12] Arlitt M, Cherkasova L, Dilley J, Friedrich R, Jin T. Evaluating content management techniques for web proxy caches. *ACM SIGMETRICS Performance Evaluation Review*, 2000, 27(4): 3-11.

[13] Wu D, Hou Y T, Zhu W, Zhang Y Q, Peha J M. Streaming video over the Internet: Approaches and directions. *IEEE Transactions on Circuits and Systems for Video Technology*, 2001, 11(3): 282-300.

[14] Li Z, Cao J, Chen G. ContinuStreaming: Achieving high playback continuity of gossip-based peer-to-peer streaming. In *Proc. the 22nd IEEE IPDPS*, Apr. 2008.

[15] Liu F, Shen S, Li B, Li B, Yin H, Li S. Novasky: Cinematic-quality VoD in a P2P storage cloud. In *Proc. the 30th INFOCOM*, Apr. 2011, pp.936-944.

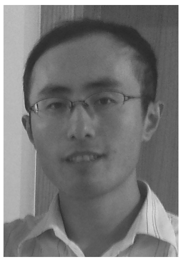
[16] Yin H, Liu X, Zhan T, Sekar V, Qiu F, Lin C, Zhang H, Li B. Design and deployment of a hybrid CDN-P2P system for live video streaming: Experiences with LiveSky. In *Proc. the 17th ACM-MM*, Oct. 2009, pp.25-34.

[17] Wu C, Li B, Zhao S. On dynamic server provisioning in multichannel P2P live streaming. *IEEE/ACM Transactions on Networking*, 2011, 19(5): 1317-1330.

[18] Li Z, Huang Y, Liu G, Wang F, Liu Y, Zhang Z L, Dai Y. Challenges, designs and performances of large-scale open-P2SP content distribution. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 24(11): 2181-2191.

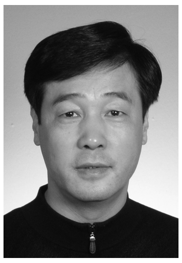
[19] Li Z, Zhang T, Huang Y, Zhang Z L, Dai Y. Maximizing the bandwidth multiplier effect for hybrid Cloud-P2P content distribution. In *Proc. the 20th ACM/IEEE IWQoS*, Jun. 2012, pp.20:1-20:9.

[20] Li Z, Wilson C, Xu T, Liu Y, Lu Z, Wang Y. Offline downloading in China: A comparative study. In *Proc. the 15th ACM IMC*, Oct. 2015.



Zhen-Hua Li is an assistant professor at the School of Software, Tsinghua University, Beijing. He received his B.S. and M.S. degrees in computer science and technology from Nanjing University in 2005 and 2008 respectively, and his Ph.D. degree in computer science and technology from Peking University in 2013. His research areas mainly consist of Internet content distribution, mobile Internet, and cloud computing/storage.

Gang Liu received his B.S. and M.S. degrees in computer science and technology from the Huazhong University of Science and Technology, Wuhan. As a system architect, he is leading the P2P and cloud computing related technologies in Tencent, particularly the QQXuanfeng cloud downloading/transcoding system.



Zhi-Yuan Ji is working at the High Technology Research and Development Center, Ministry of Science and Technology, Beijing. His research interests mainly include the scientific management of information technologies, computer application/software, and embedded systems.

Roger Zimmermann is an associate professor of computer science at the National University of Singapore (NUS). He received his Ph.D. degree in computer science and engineering from the University of Southern California in 1998. Among his research interests are mobile video management, streaming media architectures, distributed and P2P systems, spatio-temporal data management and location-based services.