Offline Downloading in China: A Comparative Study

Zhenhua Li
Tsinghua University
Tencent Co. Ltd, China
lizhenhua1983@tsinghua.edu.cn

Christo Wilson
Northeastern University
Boston, MA, US
cbw@ccs.neu.edu

Tianyin Xu
University of California
San Diego, CA, US
tixu@cs.ucsd.edu

Yao Liu
State University of New York
Binghamton University
yaoliu@binghamton.edu

Zhen Lu
Tsinghua University
Beijing, China
luzhen02@gmail.com

Yinlong Wang
Tsinghua University
Tencent Co. Ltd, China
wangyinlong239@gmail.com

ABSTRACT

Although Internet access has become more ubiquitous in recent years, most users in China still suffer from low-quality connections, especially when downloading large files. To address this issue, hundreds of millions of China's users have resorted to technologies that allow for "*offline downloading*", where a proxy is employed to pre-download the user's requested file and then deliver the file at her convenience.

In this paper, we examine two typical implementations of offline downloading: the *cloud-based approach* and the *smart AP* (access point) *based approach*. Using a large-scale dataset collected from a major cloud-based system and comprehensive benchmarks of popular smart APs, we find that the two approaches are complementary while also being subject to distinct performance bottlenecks. Driven by these results, we design and implement a proof-of-concept middleware called ODR (Offline Downloading Redirector) to help users get rid of performance bottlenecks.

We feel that offline downloading has broad applicability to other areas of the world that lack broadband penetration. By deploying offline downloading technologies, coupled with our proposed ODR middleware, the Internet experiences for users in many parts of the world can be improved.

Categories and Subject Descriptors

C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design—*Store and forward networks*; C.2.5 [**COMPUTER-COMMUNICATION NETWORKS**]: Local and Wide-Area Networks—*Internet*; C.4 [**PERFORMANCE OF SYSTEMS**]: Design studies

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Internet; offline downloading; DTN; cloud storage; smart AP

IMC'15, October 28–30, 2015, Tokyo, Japan.

DOI: http://dx.doi.org/10.1145/2815675.2815688.

1. INTRODUCTION

Although Internet access has become more ubiquitous in recent years, many users still suffer from low-quality (*i.e.,* low-bandwidth, intermittent, or restricted) network connections [54, 48, 71, 61, 51, 70]. In particular, there is a huge gap of high-speed, fixed broadband penetration between developed and developing countries. By the end of 2014, the penetration rate of fixed broadband access in the developed world has reached 27.5% with as high as 25 Mbps of download bandwidth [18]. On the other hand, the broadband penetration rate is merely 6.1% in the developing world with relatively limited, unstable download bandwidth [25].

This digital divide prevents many Internet users in developing countries from accessing the full wealth of data and services available online, especially those *large files* (*e.g.,* HD videos and large software) which require high-quality connections to download [45]. Researchers have studied various approaches to making the Internet more accessible, performant, and affordable, such as delay-tolerant networking (DTN) [17, 60]. Nevertheless, to date these technologies have not been widely deployed or evaluated in practice.

Modern China exemplifies both the promise and challenges of increasing Internet penetration [16]. In the last ten years, 46% of China's population has come online [36], and China is now home to world-class Internet companies like Tencent, Baidu, Alibaba, and Sina Weibo. However, the majority (over 72%) of China's Internet users have low-quality connections [38], due to low access bandwidth, unreliable/unstable data connection, and poor inter-connectivity between ISPs (a well-known problem in China known as the *ISP barrier*). The disparity between those who have access to high-speed, fixed broadband and those who do not is likely to increase over the next few years as more of China's rural population comes online.

To deal with the problems caused by low-quality Internet connections, hundreds of millions of China's users have resorted to technologies that allow for "*offline downloading*" of large files [44, 59, 42, 41, 10, 22, 1]. Offline downloading implements ideas from DTNs by outsourcing long downloads to a proxy, as demonstrated in Figure 1. Specifically, when a user wants to acquire a file, the user first *requests* a proxy to *pre-download* the file on her behalf (typically using an HTTP/FTP/P2P link via a low-quality network connection). The proxy may have access to faster, cheaper, or more reliable connectivity than the user, so it is better suited to downloading the file from the Internet. The user can then *fetch* the file from the proxy at a later point in time, when local network conditions are conducive to the task.

In this paper, we examine two implementations of offline downloading that are extremely popular in China [31, 33] (§ 2):

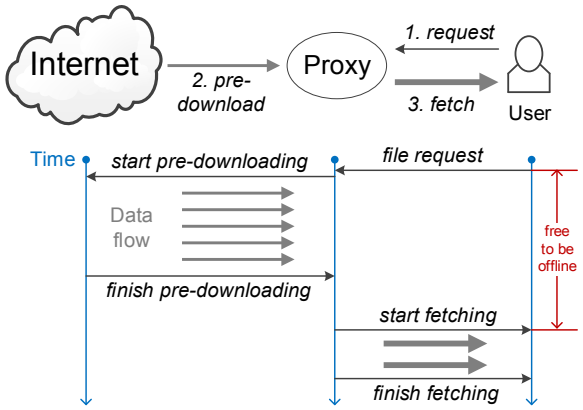

Figure 1: Basic working principle of offline downloading, as well as the corresponding timing diagram.

- *The cloud-based approach* leverages a geo-distributed, massive cloud storage pool as the proxy [59, 44, 72], which usually caches petabytes of files in a datacenter that is within or directly peered with the requesting user's ISP. This approach is adopted by Tencent Xuanfeng [41], Baidu CloudDisk [10], and Xunlei [42]. As it requires expensive cloud infrastructure for data storage and delivery, it is mostly operated by big Internet companies like Tencent and Baidu.

- *The smart AP based approach* relies on a local smart WiFi AP (access point, also known as home router) as the proxy. It is adopted by HiWiFi [21], MiWiFi [28], Newifi [30], *etc*. A traditional AP only forwards data for its connected end devices like laptops and smartphones. In contrast, a smart AP, if requested, also pre-downloads and caches data on an embedded/connected storage device (*e.g.,* an SD card, a USB flash drive, or a hard disk drive). The user can then fetch the requested file at her convenience once it is successfully pre-downloaded. This approach incurs almost zero infrastructure cost, but requires smart AP devices with redesigned hardware, OS, and application software.

Currently, there is a dispute over which approach better suits the best-effort Internet [3, 24]. This *selection dilemma* confuses the users of offline downloading services, especially those who have little expertise in Internet content delivery. The key argument lies in the performance of the two approaches, in terms of (pre-)downloading *speed*, *delay*, and *failure ratio*. Both approaches have advocates that express complaints in news media and marketing reports. However, all these disputes are either limited to one particular offline downloading service/product [59, 44, 6, 7, 8], or rely on oversimplified workloads and network environments [72, 39, 23, 4]. The former make it hard to form a general and unified view of the factors that may affect offline downloading performance, while the latter do not present comprehensive or objective results.

In this paper, we address the issue in a *quantitative* and *comparative* manner. First, we measure the workload characteristics of offline downloading (§ 3), based on a large-scale dataset collected from a major cloud-based system called Xuanfeng [41]. Our analysis of this dataset provides useful insights on the optimization of offline downloading services. Second, we identify the key performance bottlenecks of both approaches, based on the complete running logs of Xuanfeng (§ 4) and comprehensive benchmarks of three popular smart APs (§ 5). Some of our key results include:

1. *The users' fetching processes are often impeded.* The cloud-based approach can usually accelerate downloading by 7∼11 times, but performs *poorly* (*i.e.,* the user's fetching speed falls below 1 Mbps, or 125 KBps, and is thus unfit for HD-video streaming) once there is a bandwidth bottleneck in the network path between the cloud and the user. Specifically, a high portion (28%) of Xuanfeng users suffer from this bottleneck, which is mainly caused by cross-ISP data delivery, low user-side access bandwidth, or lack of cloud-side upload bandwidth. These users should utilize an additional smart AP to mitigate these impediments.

2. *The cloud's upload bandwidth is being overused.* The cloud-based approach is threatened by running out of its upload bandwidth due to *unnecessarily* sending *highly popular* files. A small percentage (0.84%) of highly popular files account for a large part (39%) of all downloads. As the user base grows, the cloud of Xuanfeng will have to reject more (>1.5%) users' fetching requests. Because the majority (87%) of requested files are hosted in peer-to-peer data swarms [47], many highly popular files can be directly downloaded by users with as good or greater performance than what is provided by cloud-based offline downloading services.

3. *Smart APs frequently fail during pre-downloading.* Although smart APs are immune to the above performance bottlenecks of the cloud-based approach, they frequently (42%) fail while pre-downloading *unpopular* files. This is mainly caused by insufficient seeds in a P2P data swarm and poor HTTP/FTP connections. Note that 36% of offline downloading requests are issued for unpopular files. Therefore, users who need to download unpopular files should choose the cloud-based approach, which is better at downloading unpopular files — the failure ratio (13%) is much lower owing to *collaborative caching* (refer to § 2.1) in the massive cloud storage pool.

4. *Smart APs can be slower due to hardware/filesystem restrictions.* Surprisingly, a smart AP's pre-downloading speed can be restricted by its hardware and filesystem. This is because some types of storage devices (*e.g.,* USB flash drive) and filesystems (*e.g.,* NTFS) do not fit the pattern of frequent, small data writes during the pre-downloading process. These smart APs would benefit from upgraded storage devices and/or alternate filesystems, so as to release the full potential of offline downloading.

Driven by these results, we design and implement a proof-of-concept middleware called ODR (Offline Downloading Redirector, available at *http://odr.thucloud.com*) to help users get rid of performance bottlenecks (§ 6). As illustrated in Figure 2, the basic idea of ODR is to adaptively redirect the user's file request to where the best performance is expected to be achieved, including the cloud (we use the Xuanfeng cloud in our implementation), the smart AP, the user's local device, or a combination. ODR's primary goal is to minimize the downloading time and failure ratio; its secondary goal is to minimize the upload bandwidth burden on the cloud.

ODR makes redirection decisions based on two types of information. First, after receiving an offline downloading request, ODR queries the content database of Xuanfeng to obtain the popularity information of the requested file. Then, ODR examines whether there is a potential bandwidth bottleneck by analyzing the user's IP address, access bandwidth, storage device, and so forth. Note that ODR requires no modification to existing cloud-based systems or smart AP devices. As a result, it is easy to deploy in practice.

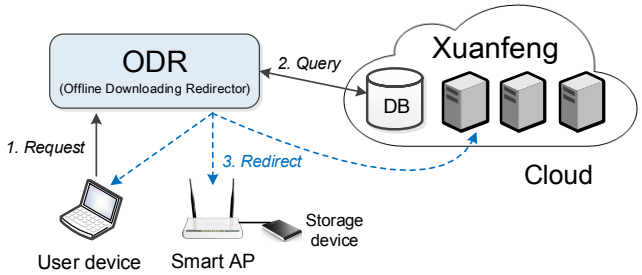

Figure 2: Basic working principle of ODR.

To validate the efficacy of ODR, we replay an unbiased sample of Xuanfeng users' offline downloading requests using a prototype ODR system. The evaluation results indicate that ODR is able to effectively overcome the performance bottlenecks. First, the percentage of impeded fetching processes is reduced from 28% to 9%. Second, the cloud's upload bandwidth burden is reduced by 35% and thus no fetching request will need to be rejected. Third, the percentage of smart APs' failures in pre-downloading unpopular files is reduced from 42% to 13%. Last, the hardware/filesystem restrictions on smart APs' pre-downloading speeds are almost completely avoided.

Finally, we feel that offline downloading has broad applicability to other areas of the world that lack broadband penetration. By deploying offline downloading technologies, coupled with our proposed ODR middleware, the Internet experiences for users in many parts of the world can be significantly improved.

2. SYSTEM OVERVIEW

This section provides an overview of the systems studied in this paper, including the cloud-based offline downloading system (Xuanfeng) and three popular smart AP systems.

2.1 Overview of Xuanfeng

Xuanfeng is a major provider of cloud-based offline downloading service in China, possessing over 30 million users at the moment (including a small portion of overseas users). Its service can be accessed via either a PC client (available from *http://xf.qq.com*) or a web-based portal (available at *http://lixian.qq.com/main.html*). The former access method is dominant due to its full-fledged functionality (supporting almost all the common file transfer protocols like HTTP/FTP and BitTorrent/eMule). In terms of business model, Xuanfeng exists as a value-added service of the Tencent company, *i.e.,* any registered Tencent user can access it freely. Baidu CloudDisk [10] and Xunlei [42] are the main competitors of Xuanfeng [1]. The former is also a free service, while the latter charges its users around $1.50 per month.

As depicted in Figure 3, the system architecture of the Xuanfeng cloud mainly consists of three clusters of servers: 1) *pre-downloading servers*, 2) *storage servers*, and 3) *uploading servers*, as well as a database (DB) for maintaining the metadata information of users and cached files. The total cloud storage space (spread across nearly 500 commodity servers) is nearly 2 PB at the moment, caching around 5 million files. The cached files are replaced in an LRU (least recently used) manner.

[1]Note that the configuration and engineering of Xunlei and Baidu CloudDisk may differ from those of Xuanfeng. Hence, their performance bottlenecks may also be different.

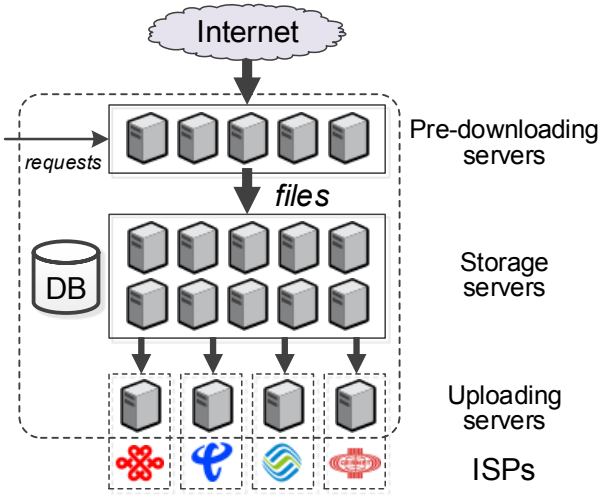

Figure 3: System architecture of the Xuanfeng cloud.

Collaborative caching. In the Xuanfeng cloud, all users' requested files are cached in a *collaborative* way. Specifically, every file is identified using the MD5 hash code of its content, which facilitates file-level deduplication across different users. Consequently, the vast majority (89%) of offline downloading requests can be instantly satisfied with cached files and then no pre-downloading bandwidth cost is incurred. Xuanfeng does not utilize chunk-level deduplication to avoid trading high chunking complexity for low (<1%) storage space savings. The low storage savings come from the fact that there do exist a few videos sharing a portion of frames/chunks.

When a user-requested file cannot be found in the cloud cache, Xuanfeng assigns a virtual machine (named a *pre-downloader*) to pre-download the file from the Internet. The Internet access bandwidth of a pre-downloader is around 20 Mbps (= 2.5 MBps), equivalent to the high-end, fixed broadband bandwidth in China.

Privileged network path. China has a different Internet topology from Europe and the US. Particularly, China has a simple AS topology with a small number of major ISPs; each ISP has a giant AS built on top of a nationwide backbone network [68]. As a consequence, the performance of cross-ISP data delivery is significantly degraded, a problem which is known as the *ISP barrier* [69, 49]. To accelerate users' fetching processes, Xuanfeng tries to construct *privileged network paths* between the cloud and users by deploying uploading servers within the four major ISPs: Unicom [15], Telecom [14], Mobile [13], and CERNET [12].

To construct a privileged network path, Xuanfeng always attempts to select an uploading server that resides in the same ISP as the fetching user, so that the ISP barrier is avoided. However, privileged path construction may fail in two cases: 1) The fetching user is not within any of the four major ISPs; 2) The fetching user is within one of the four major ISPs (say CERNET) but the uploading servers in CERNET have exhausted their upload bandwidth at that time. In either case, Xuanfeng would select an alternative uploading server that has the shortest network latency from the user.

Once a privileged network path is set up, Xuanfeng sets no limitation on the user's fetching speed, with maximum speeds reaching 50 Mbps (= 6.25 MBps). However, at some "peak" point, all the uploading servers may have exhausted their upload bandwidth. In this case, Xuanfeng temporarily rejects new fetching requests rather than degrade the speeds of active downloads.

Table 1: Hardware configurations of the three popular smart APs studied in this paper.

Smart AP	CPU	RAM	Storage Interface (and Device)	WiFi Protocol and Channel
HiWiFi (1S)	MT7620A @580 MHz	128 MB	an SD card interface	IEEE 802.11 b/g/n @2.4 GHz
MiWiFi	Broadcom4709 @1 GHz	256 MB	a USB 2.0 interface and an internal 1-TB SATA hard disk drive	IEEE 802.11 b/g/n/ac @2.4/5.0 GHz
Newifi	MT7620A @580 MHz	128 MB	a USB 2.0 interface	IEEE 802.11 b/g/n/ac @2.4/5.0 GHz

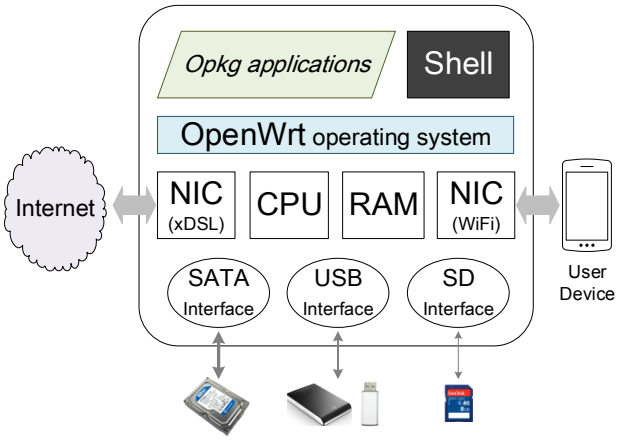

Figure 4: System architecture of a commerical smart AP. Note that a given smart AP device may not contain all potential storage device interfaces.

2.2 Overview of the Smart AP Systems

Though smart APs have only been on the market for around two years, they have quickly gained enormous popularity in China. HiWiFi, the first widely available smart AP device released in Mar. 2013 [20], is now striving towards 5 million sales [22]. Despite the late entry into the market (in May 2014), 100,000 MiWiFi devices were sold out in just 59 seconds [1].

As depicted in Figure 4, the system architecture of popular smart APs is essentially made up of three parts: 1) Basic hardware, including CPU, RAM, NIC (network interface card) for xDSL, NIC for WiFi, and the storage device interface(s); 2) The operating system (typically OpenWrt [34]); 3) Shell and the other applications.

Table 1 lists the hardware configurations of the three popular smart APs we examine in this study. Among them, MiWiFi has the best configuration in terms of CPU, RAM, storage device, and WiFi data transfer. Accordingly, MiWiFi costs more than $100, while HiWiFi and Newifi each costs around $20.

All the three popular smart APs utilize OpenWrt, a Linux-based embedded operating system. OpenWrt provides a fully writable filesystem and supports the lightweight Opkg package management system [35], which allows users to install any Opkg application. For instance, the three APs make use of wget and aria2 [9] to support HTTP/FTP and BitTorrent/eMule protocols. Also, a web interface is provided for users to specify offline downloading requests. In our benchmark experiments, we measure and record detailed performance of smart APs with Opkg tools/apps such as BASH, tcpdump, top, iostats, and scp.

3. WORKLOAD CHARACTERISTICS

To understand the workload characteristics of offline downloading, we study a large-scale dataset collected from Xuanfeng. We assume that the smart AP based offline downloading systems have similar workload characteristics to Xuanfeng, since most end users are not familiar with the technical details and cannot differentiate these services. The large-scale dataset contains the complete running logs of Xuanfeng during a whole week (Feb. 22–28, 2015), involving 4,084,417 offline downloading tasks, 783,944 users, and 563,517 unique files. Corresponding to the three stages of offline downloading (refer to Figure 1), the dataset is composed of the following three parts:

1. The trace of user requests (workload trace) records all the offline downloading requests issued by users. For each request, the logs record the user ID, IP address, access bandwidth (if available), request time, file type, file size, link to the original data source, and file transfer protocol;

2. The pre-downloading trace records the performance data of the proxy's pre-downloading user-requested files. It includes the start time, finish time, acquired file size, network traffic consumed, cloud cache hit status, average downloading speed, peak downloading speed, and success or failure for each pre-downloading process;

3. The fetching trace records the performance data of users' fetching processes from the proxy. It contains the user ID, IP address, access bandwidth (if available), start time, finish/pause time, acquired file size, network traffic consumed, average fetching speed, and peak fetching speed for each fetching process.

This section studies the first part of the dataset (*i.e.,* users' offline downloading requests). The second and third parts of the dataset will be examined in § 4 and § 5.

File type. In the workload trace, the majority (75%) of offline downloading requests are issued for videos; the other 15% are for software packages. The reason is intuitive: among all the requested files, videos are the largest in size, *i.e.,* the most time and traffic consuming to download, so users are more inclined to issue offline downloading requests for videos. This suggests that offline downloading systems should be mainly optimized for videos.

File size. As shown in Figure 5, the average size of requested files is 390 MB and the maximum size is 4 GB, which is consistent with our observation that most requested files are large videos. Nevertheless, we also find that up to 25% of requested files are smaller than 8 MB in size, most of which are demo videos, pictures, documents, and small software packages.

File transfer protocol. The majority (87%) of requested files are hosted in P2P data swarms, including BitTorrent (68%) and eMule (19%) swarms. The remaining 13% are hosted on HTTP or FTP servers. Thus, designers of offline downloading systems should pay special attention to P2P-based file transfer protocols.

Since P2P suffers from several technical shortcomings (*e.g.,* high dynamics and heterogeneities among end-user devices and network connections), it is often difficult for users to find peers (including both seeds and leechers) sharing the target files. As a result, the downloading efficacy of P2P can be poor and unpredictable. For

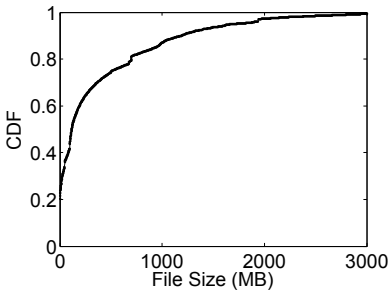

Figure 5: CDF of requested file size. Min size: 4 B, Median size: 115 MB, Average size: 390 MB, and Max size: 4 GB.

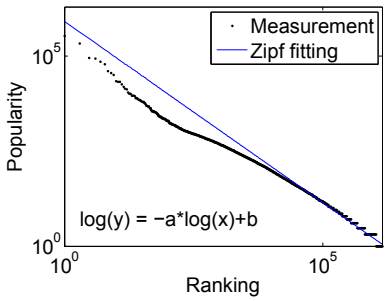

Figure 6: Popularity distribution of requested files — Zipf fitting. The average relative error of fitness is 15.3%.

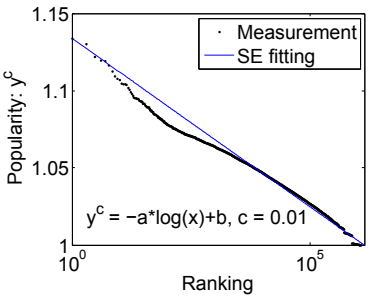

Figure 7: Popularity distribution of requested files — SE fitting. The average relative error of fitness is 13.7%.

this reason, users resort to offline downloading to obtain P2P files over the long run. On the contrary, HTTP and FTP servers are usually stable with more predictable performance, so online downloading is preferred. This explains the dominance of P2P in terms of file transfer protocol.

File popularity. As mentioned in § 2.1, Xuanfeng actively maintains a content database where every file is associated with a unique identifier (ID). The ID is the MD5 hash code of the complete file content. Hence, files are considered *identical* as long as they have the same ID. Figure 6 (in $\log(x)$ *vs.* $\log(y)$ style) and Figure 7 (in $\log(x)$ *vs.* y^c style) indicate that the popularity distribution of requested files is highly skewed, approximately following the well known Zipf model [46] or the SE (stretched exponential) model [56].

Let x denote the popularity ranking of a file, and y denote its popularity (according to the workload trace). Then, we have the following fitting equations:

$$\text{Zipf:} \quad log(y) = -a_1 \times log(x) + b_1,$$
$$\text{SE:} \quad y^c = -a_2 \times log(x) + b_2,$$

where $a_1 = 1.034$, $b_1 = 14.444$, $a_2 = 0.010$, $b_2 = 1.134$, and $c = 0.01$.

With regard to the average relative error of fitness, SE (13.7%) seems to be a better fit than Zipf (15.3%), especially for those small-ranking (*i.e.,* most popular) files. The reason why SE fits the measurement data better than Zipf can be mainly attributed to the *fetch-at-most-once* effect of P2P video files [55]. It is known that 75% of offline downloading requests are issued for videos and 87% of requested files are hosted in P2P data swarms. That is to say, P2P video files dominate the workload of Xuanfeng. Specifically, a given Xuanfeng user generally fetches a P2P video file for at most once, whereas web pages and other small documents are often fetched repeatedly [46]. Therefore, the access pattern of offline downloaded files deviates from the Zipf access pattern of web objects. The above finding complements the analysis of file popularity in previous studies of offline downloading [59, 72], which simply used the Zipf model to characterize the workload.

4. PERFORMANCE OF THE CLOUD-BASED SYSTEM

This section presents the performance of our representative cloud-based offline downloading system Xuanfeng, including both the pre-downloading and fetching phases. In addition, we analyze the end-to-end performance by combining the two phases.

4.1 Pre-downloading Performance

Pre-downloading speed. Figure 8 (the upper curve) plots the distribution of pre-downloading speeds in Xuanfeng. The median speed is merely 25 KBps, which indicates that half of the pre-downloading processes are quite slow. The average speed (69 KBps) is higher than the median speed, but is still far from satisfactory — keep in mind that, as mentioned in § 2.1, the access bandwidth of a pre-downloader is around 20 Mbps (= 2.5 MBps). 21% of the pre-downloading processes even have a speed close to 0 KBps, which is mostly caused by pre-downloading failures.

Pre-downloading delay. Figure 9 (the lower curve) plots the distribution of the pre-downloading delay in Xuanfeng (excluding the cache-hit cases where the pre-downloading delay is zero). The average delay is as high as 370 minutes, which is much longer than the length of a common movie (100∼120 minutes). Obviously, such long delay is unfit for continuous video streaming [63, 53]. Also, the average delay is much longer than the median delay (82 minutes), indicating that the pre-downloading delay of many requested files is extremely long.

Failure ratio. A *pre-downloading failure* occurs when the service gives up the pre-downloading attempt and notifies the requesting user of the failure. It is hard to theoretically define a failure if we allow the pre-downloading process to take infinite time. However, practical systems have to timeout the pre-downloading process if the expected completion time is not reasonable. In practice, Xuanfeng raises a pre-downloading failure for a requested file when *the corresponding pre-downloading progress stagnates for an hour*. This timeout rule is supported by our following observation in Xuanfeng: if the pre-downloading progress of a requested file stagnates for an hour, then this file can hardly be successfully pre-downloaded even if the timeout threshold is set to be one week.

The overall pre-downloading failure ratio of Xuanfeng is 8.7%. Note that if a user-requested file is already cached in the cloud storage pool, the pre-downloading is immediately successful. On the other hand, if we do not take the cache hit cases into account (*i.e.,* we assume that the cloud storage pool does not exist), the overall pre-downloading failure ratio will increase to 16.4%. This confirms the importance of the massive cloud storage pool as well as the collaborative caching mechanism (refer to § 2.1) in mitigating pre-downloading failures.

More importantly, we discover that the pre-downloading failure ratio correlates with the popularity of requested files, as shown in the scatter plot of Figure 10. For the purposes of this discussion, we define a file to be *unpopular* if it was downloaded less than 7 times per week. *Popular* files are downloaded 7–84 times per

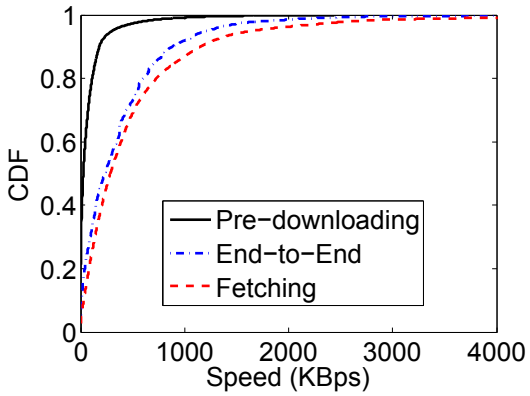

Figure 8: CDF of pre-downloading and fetching speeds in Xuanfeng. As for the pre-downloading speed (excluding the cache-hit cases), Min: 0 KBps, Median: 25 KBps, Average: 69 KBps, and Max: 2.37 MBps (≈ 20 Mbps). As for the fetching speed, Min: 0 KBps, Median: 287 KBps, Average: 504 KBps, and Max: 6.1 MBps (≈ 50 Mbps). As for the end-to-end speed, Min: 0 KBps, Median: 233 KBps, Average: 380 KBps, and Max: 6.1 MBps (≈ 50 Mbps).

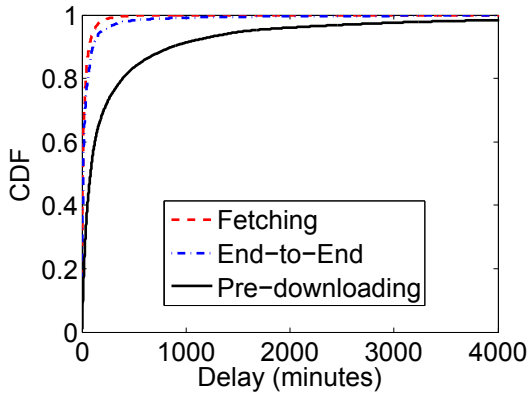

Figure 9: CDF of pre-downloading, fetching, and end-to-end delay in Xuanfeng. As for the pre-downloading delay (excluding the cache-hit cases), Min: 0 minute, Median: 82 minutes, Average: 370 minutes, and Max: 10071 minutes. As for the fetching delay, Min: 0 minute, Median: 7 minutes, Average: 27 minutes, and Max: 9724 minutes. As for the end-to-end delay, Min: 0 minute, Median: 10 minutes, Average: 68 minutes, and Max: 19553 minutes.

week, while *highly popular* files are downloaded greater than 84 times per week. Given these definitions, we find that 93.2% of files are unpopular while only 0.84% of files are highly popular. However, only 36% of offline downloading requests are issued for the 93.2% unpopular files, while 39% of requests are issued for the 0.84% highly popular files.

Network traffic cost. As discussed in § 3, only 13% of requested files are hosted in HTTP or FTP servers, while 87% are hosted in P2P data swarms. For HTTP and FTP, the pre-downloading traffic is slightly (typically 7%–10%) larger than the file size; the overhead mainly comes from HTTP, FTP, TCP, and IP packet headers. For P2P, its "tit-for-tat" policy [50] (*i.e.,* a peer that downloads data from others must upload data to others at the same time) causes the pre-downloading traffic to be considerably (50%–150%) larger than the file size. In the Xuanfeng system, we observe that the overall pre-downloading traffic is 196% of the total file size. That is to say, the overhead traffic is comparable to the file size.

4.2 Fetching Performance

Fetching speed. Figure 8 (the lower curve) shows the distribution of users' fetching speeds from the cloud. Owing to the privileged network paths constructed by Xuanfeng (refer to § 2.1), the median and average fetching speeds are as high as 287 KBps and 504 KBps. Given that the median and average pre-downloading speeds are merely 25 KBps and 69 KBps, Xuanfeng greatly improves *perceived* downloading speeds (by 7∼11 times in terms of median and average speeds) for China's Internet users.

When videos are fetched from the proxy, Xuanfeng supports two different modes: a) *view-as-download* and b) *view-after-download*. According to Xuanfeng users' behaviors, most users tend to choose the former mode over the latter. Therefore, to facilitate users' real-time video playback in a continuous manner, a *bandwidth bottleneck* is recognized when the fetching speed falls below 125 KBps. The 125 KBps threshold corresponds with the typical 1 Mbps playback rate of large (HD) videos [57, 62].

Specifically, a high portion (28%) of fetching speeds are below 125 KBps. To explore why Xuanfeng exhibits poor performance

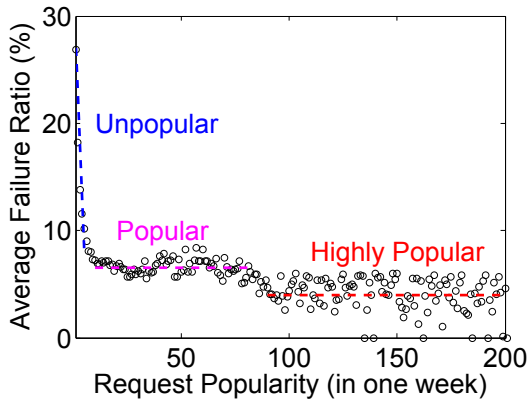

Figure 10: Request popularity *vs.* Pre-downloading failure ratio. Request popularity: [0, 7) → unpopular files, [7, 84] → popular files, and (84, MAX] → highly popular files.

during these fetching processes, we carefully examine the corresponding logs in the fetching trace, and find that the bandwidth bottleneck is mainly caused by three issues: 1) cross-ISP data delivery, 2) low user-side access bandwidth, and 3) lack of cloud-side upload bandwidth.

In detail, 9.6% of fetching processes are limited by the ISP barrier. For these fetching processes, the users' IP addresses do not belong to any of the four ISPs supported by Xuanfeng. Besides, around 10.8% of fetching processes are limited by low user-side access bandwidth [2] (< 125 KBps). Additionally, there are 1.5% of fetching requests rejected by the cloud due to lack of upload bandwidth (refer to § 2.1). These rejected requests explain why the minimum observed fetching speed is 0 KBps. Finally, the remaining unmentioned portion ($= 28\% - 9.6\% - 10.8\% - 1.5\% = 6.1\%$) are owing to unknown reasons we have not figured out yet, possibly because of the network dynamics or system bugs.

[2]Some users of Xuanfeng did not report their access bandwidth. For these users, we use the peak fetching speed recorded in the fetching trace to approximate their access bandwidth.

Fetching delay. Figure 9 (the upper curve) shows the distribution of users' fetching delay. As a consequence of most users' high fetching speeds, the median fetching delay is as low as 7 minutes and the average fetching delay is merely 27 minutes. Given that the median and average pre-downloading delay is as high as 82 minutes and 370 minutes, Xuanfeng has significantly shortened *perceived* downloading delay (by 12∼14 times in terms of median and average fetching delay) for China's Internet users.

In summary, we find the first key performance bottleneck of offline downloading:

- *The cloud-based approach performs poorly once there is a bandwidth bottleneck in the privileged network path between the cloud and the user. This bottleneck is mainly caused by cross-ISP data delivery, low user-side access bandwidth, or lack of cloud-side upload bandwidth.*

Shortage of cloud bandwidth. In order to support high-speed fetching from the cloud to users' devices, Xuanfeng has purchased a total of 30 Gbps of upload bandwidth from the four major ISPs in China. In the 7th day of the measurement week, the peak cloud-side upload bandwidth burden exceeded 30 Gbps, as demonstrated in Figure 11. Consequently, a small portion (1.5%) of fetching requests were rejected by Xuanfeng. As the user base continues to grow, Xuanfeng will have to reject more fetching requests, which unfortunately harms the user experience.

On the other hand, if we look deeply into the usage of cloud bandwidth, we find that *the current cloud-side upload bandwidth burden is not all necessary.* We calculate the cloud-side bandwidth used for delivering (uploading) highly popular files based on the fetching trace, plotted as the lower, blue curve in Figure 11. We can see that, on average, nearly 40% of the cloud bandwidth is spent delivering highly popular files. The reason is that a small percentage (0.84%) of highly popular files account for a large part (39%) of all downloads, as discussed in § 4.1.

In fact, because the majority (87%) of requested files are hosted in P2P data swarms, many highly popular files can be directly downloaded by users with as good or greater performance than what the cloud provides [66, 64]. This is because P2P data sharing among end users can achieve the so-called "*bandwidth multiplier effect*" [66]. Specifically, by appropriately allocating a certain portion of cloud bandwidth (S_i) to a P2P data swarm i to seed the content, the Xuanfeng system can attain a higher aggregate content distribution bandwidth (D_i) by letting P2P users exchange data and distribute content among themselves. The ratio $\frac{D_i}{S_i}$ is referred to as the bandwidth multiplier for P2P data swarm i. The above observation and analysis lead to the second key performance bottleneck of offline downloading:

- *The cloud-based approach is threatened by running out of upload bandwidth due to unnecessarily sending highly popular files. As the user base continues to grow, the cloud will have to reject more (>1.5%) fetching requests.*

User-side network overhead. Xuanfeng minimizes the user-side network overhead, as its users only download requested files from the cloud and do not upload data to others. On average, a user's downloading traffic usage is slightly (7% – 10%) larger than the file size. This is especially useful for mobile P2P users with limited data plans or traffic caps. As mentioned in § 4.1, for an average P2P user to download a file from the corresponding data swarm, the total traffic usage is 196% of the total file size. Therefore, by resorting to Xuanfeng rather than the original data swarm, an average P2P user could achieve considerable traffic usage saving which is comparable to 86% – 89% of the total file size.

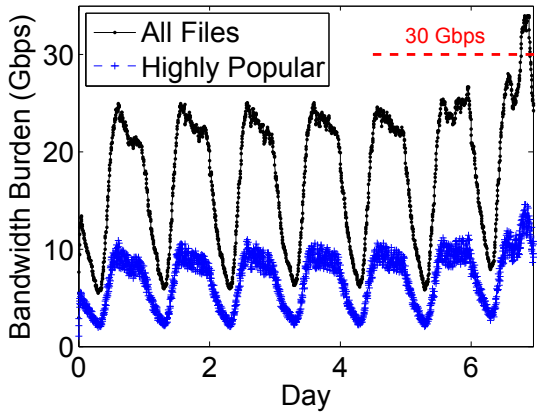

Figure 11: Cloud-side upload bandwidth burden of Xuanfeng in the measurement week (including those rejected fetching requests). The time interval is 5 minutes. Note that for the 1.5% of fetching requests rejected by Xuanfeng in the 7th day, their incurred cloud-side upload bandwidth burden (which did not really happen) is estimated by approximately taking their average fetching speed as 504 KBps (*i.e.,* , the average speed of all the real fetching processes in Xuanfeng, refer to Figure 8).

4.3 End-to-End Performance

Finally, we analyze the *end-to-end speed and delay* of offline downloading in Xuanfeng. For a complete offline downloading process, the end-to-end delay is the sum of pre-downloading delay and fetching delay, and the end-to-end speed is the size of the requested file divided by the end-to-end delay.

As illustrated in Figure 9, the CDF curve of end-to-end delay falls between the CDF curves of pre-downloading delay and fetching delay, which is within our expectation. More importantly, the distribution of end-to-end delay looks much closer to the distribution of fetching delay. This is because the vast majority (89%) of offline downloading requests can be instantly satisfied by the cloud cache and thus the corresponding pre-downloading delay is almost zero (refer to § 2.1). Similarly, Figure 8 indicates that the CDF curve of end-to-end speed falls between the CDF curves of pre-downloading speed and fetching speed, and the distribution of end-to-end speed is much closer to the distribution of fetching speed.

From the above observations, we conclude that even from an end-to-end perspective, the offline downloading service provided by Xuanfeng can effectively improve the *perceived experiences* for its users.

5. PERFORMANCE OF THE SMART APS

Now that we understand the dynamics and performance characteristics of Xuanfeng, we move on to examining smart AP based offline downloading. In this section, we report our benchmark methodology and the performance of the three most popular smart APs in China: HiWiFi, MiWiFi, and Newifi.

5.1 Methodology

To comprehensively measure the performance of the three popular smart APs (during Mar. 1–22, 2015), we randomly sample 1000 real offline downloading requests [3] issued by Unicom users

[3]Each selected request record should contain the user's access bandwidth information so that we can approximate the user's real network connection characteristics during our benchmarks.

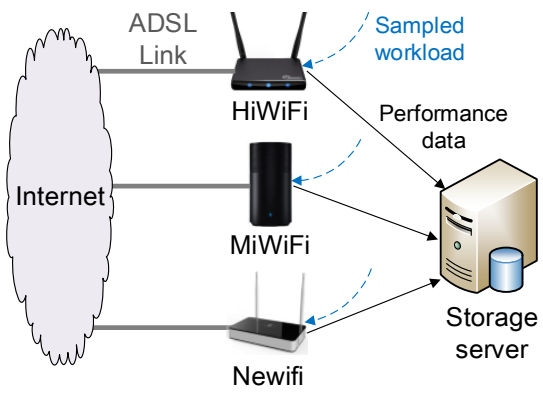

Figure 12: Our benchmark environment. For each smart AP, its embedded/connected storage device is not plotted to make the figure tidy.

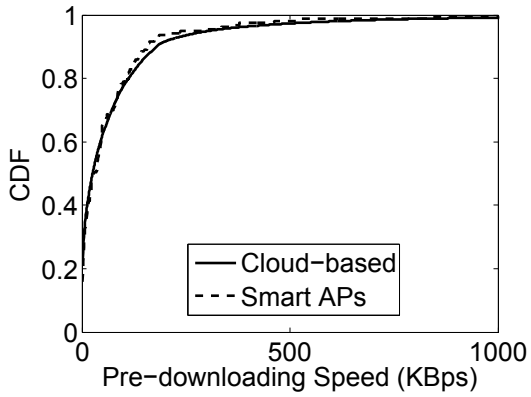

Figure 13: CDF of smart APs' pre-downloading speeds. Min: 0 KBps; Median: 27 KBps; Average: 64 KBps; Max: 2.37 MBps (≈ 20 Mbps) for HiWiFi and MiWiFi, and 0.93 MBps for Newifi. As a comparison, the CDF of cloud-based pre-downloading speeds is also plotted.

in the workload trace of Xuanfeng (refer to § 3). These sampled requests (which we refer to as the sampled workload) are restricted to Unicom users because our benchmark experiments are conducted by replaying the sampled workload with smart APs on Unicom network connections. On the other hand, the data sources of these sampled requests are located across various ISPs in China. For every request record, we ignore its user ID, IP address, and request time (since these factors cannot be reproduced in our benchmarks), but reuse the user's access bandwidth, as well as the file type, file size, link to the original data source, and file transfer protocol.

To replay the sampled workload using the three smart APs, we utilize three independent residential ADSL links provided by the Unicom ISP, each of which was used exclusively by one smart AP, as depicted in Figure 12. Each link has 20 Mbps (= 2.5 MBps) of Internet access bandwidth. When replaying an individual offline downloading request, we restrict the smart AP's pre-downloading speed within the recorded user access bandwidth, so as to approximate the real network connection status. We sequentially replay around 333 requests (request $i+1$ is replayed after request i completes or fails) on each smart AP and record the performance data.

HiWiFi uses an embedded 8-GB SD card (Max Write/Read Speed: 15 MBps/30 MBps) as the storage device. The SD card can only be formatted as FAT (otherwise, HiWiFi does not work). Newifi uses an external 8-GB USB flash drive (Max Write/Read Speed: 10 MBps/20 MBps) via a USB 2.0 interface. The USB flash drive is formatted as NTFS. Since both HiWiFi and Newifi have a small storage capacity, we remove requested files from the storage device after they are completely downloaded or the corresponding pre-downloading task failures (refer to § 4.1). At the same time, the performance data is aggregated into a storage server. MiWiFi uses its internal 1-TB SATA hard disk drive (5400 RPM, Max Write/Read Speed: 30 MBps/70 MBps). This hard disk drive has been formatted as EXT4 by the manufacturer, and it cannot be re-formatted to any other filesystem.

5.2 Benchmark Results

Since smart APs are located in the same LAN (local area network) as users, the performance of the fetching phase is seldom an issue except when multiple user devices are fetching from a smart AP at the same time. Specifically, a user can fetch from a smart AP by directly dumping from the AP's storage device or through a wired/WiFi LAN connection. Even the lowest WiFi fetching speed lies in 8∼12 MBps, which is higher than the maximum fetching speed (*i.e.,* 6.1 MBps) of Xuanfeng users. As a consequence, below we focus on the performance of the pre-downloading phase.

Pre-downloading failure ratio. The overall pre-downloading failure ratio of smart APs is 16.8%, which is higher than that of Xuanfeng (8.7%). More importantly, for unpopular files the pre-downloading failure ratio of smart APs is as high as 42%, significantly higher than that of Xuanfeng (13%). Note that according to the workload trace of Xuanfeng, 36% of offline downloading requests are issued for unpopular files. With these results, we discover the third key performance bottleneck of offline downloading:

- *Although smart APs are immune to the two bottlenecks of the cloud-based approach described in § 4, they frequently fail in pre-downloading unpopular files. In contrast, the cloud-based approach performs much better at downloading unpopular files since it benefits from the massive geo-distributed cloud storage pool.*

Further, we delve into the details of the pre-downloading failures of smart APs. Among the 168 ($= 1000 \times 16.8\%$) failures, the vast majority (145, 86%) were caused by insufficient seeds in a P2P data swarm. This explains why the pre-downloading failure ratio of unpopular files is especially high. Besides, a non-negligible part (17, 10%) were ascribed to poor HTTP/FTP connections, *i.e.,* the server at the other end failed to maintain a persistent/resumable download. The remainder (6, 4%) might be the result of system bugs in HiWiFi, MiWiFi and Newifi.

Pre-downloading speed and delay. As shown in Figure 13 and Figure 14, the pre-downloading speeds of smart APs are just a bit lower than those of Xuanfeng's pre-downloaders, and thus the pre-downloading delay of smart APs seems a bit longer than that of Xuanfeng. This is because smart APs work in a similar way as the pre-downloaders (*i.e.,* they also download files using HTTP, FTP and P2P protocols; refer to § 2.1), although their access bandwidths are generally lower than 20 Mbps (or 2.5 MBps, *i.e.,* a pre-downloader's access bandwidth).

However, one detail in Figure 13 and Figure 14 is worth discussing: the median pre-downloading speed of smart APs (27 KBps) is *higher* than that of Xuanfeng (25 KBps), but the average pre-downloading speed of smart APs (64 KBps) is *lower* than that of Xuanfeng (69 KBps). Accordingly, the median pre-downloading

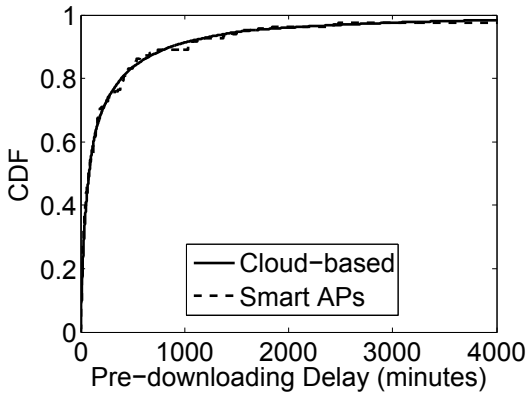

Figure 14: CDF of smart APs' pre-downloading delay. Min: 0 minutes; Median: 77 minutes; Average: 402 minutes; Max: 8297 minutes. As a comparison, the CDF of cloud-based pre-downloading delay is also plotted.

delay of smart APs (77 minutes) is *shorter* than that of Xuanfeng (82 minutes), but the average pre-downloading delay of smart APs (402 minutes) is *longer* than that of Xuanfeng (370 minutes). To demystify this counter-intuitive phenomenon, we examine the pre-downloading performance data of every task in the sampled workload and its original performance data in the pre-downloading trace. Surprisingly, we uncover the fourth key performance bottleneck of offline downloading:

- *A smart AP's pre-downloading speed can be restricted by its hardware and/or filesystem, since some types of storage devices and filesystems do not fit the pattern of frequent, small data writes during the pre-downloading process.*

In particular, among all the experimented storage devices, USB flash drive is the slowest. When Newifi uses the USB flash drive (in the NTFS format) as its storage device, the max pre-downloading speed is merely 0.93 MBps, much lower than that of HiWiFi and MiWiFi (2.37 MBps). This finding is basically consistent with Sundaresan *et al.* 's study results on broadband Internet performance in 2011, which also indicate that a user's home network equipment (including the home router of course, though smart APs did not exist in 2011) can significantly affect downloading performance [67].

To comprehensively understand the influence of hardware and filesystem on Newifi's pre-downloading speed, we replay the top-10 popular requests in the sampled workload on Newifi while setting no restriction on its pre-downloading speed (so the maximum speed should be nearly 2.37 MBps). We conducted replays with the USB flash drive formatted as FAT, NTFS, and EXT4, respectively, as well as with the USB flash drive replaced by a USB hard disk drive (5400 RPM, Max Write/Read Speed: 10 MBps/25 MBps). The resulting maximum pre-downloading speeds, as well as the corresponding iowait ratios, are listed in Table 2. We make three observations as follows:

- The NTFS filesystem severely harms Newifi's maximum pre-downloading speed, no matter whether the storage device is a USB flash drive or a USB hard disk drive. This is mainly attributed to the incompatibility between NTFS and Newifi's OpenWrt operating system (refer to § 2.2) which utilizes the EXT4 filesystem.

- When Newifi uses a USB flash drive as its storage device, FAT and EXT4 filesystems also appear to have degraded its maximum pre-downloading speed. The major reason should be the unsuitability of the USB flash drive on handling frequent, small data writes during pre-downloading. This is reflected by the high iowait ratios (66.3% and 55%) as shown in Table 2. Besides, we observe that the receiver-side TCP sliding window (the typical size is 14608 bytes) is almost full in most of the time during the pre-downloading process.

- When Newifi uses a USB hard disk drive, its maximum pre-downloading speed is considerably enhanced (compared with using a USB flash drive) even when the hard disk drive is formatted as NTFS. No matter which filesystem is used, the iowait ratio is relatively low.

Table 2: Max pre-downloading speeds and the corresponding iowait ratios for HiWiFi, MiWiFi, and Newifi, with different storage devices and filesystems.

Max pre-downloading speed (MBps)	FAT	NTFS	EXT4
HiWiFi + SD card	2.37	–	–
MiWiFi + SATA hard disk drive	–	–	2.37
Newifi + USB flash drive	**2.12**	**0.93**	**2.13**
Newifi + USB hard disk drive	2.37	**1.13**	2.37
iowait ratio	**FAT**	**NTFS**	**EXT4**
HiWiFi + SD card	42.1%	–	–
MiWiFi + SATA hard disk drive	–	–	29.7%
Newifi + USB flash drive	**66.3%**	15.1%	**55%**
Newifi + USB hard disk drive	42%	9.8%	17.4%

Based on the above observations, we suggest that Newifi-like smart APs upgrade their storage devices and/or change their default filesystems to more performant variants, so as to release the full potential of offline downloading. Currently, because Newifi only has a USB 2.0 interface (refer to Table 1), using a USB hard disk drive coupled with the EXT4 filesystem seems to be the best fit. On the other hand, if Newifi upgrades its storage interface to USB 3.0 in the future, using a USB 3.0 flash drive formatted as EXT4 might be a more cost-effective choice, given that a USB flash drive is usually much smaller and cheaper than a USB hard disk drive.

6. THE ODR MIDDLEWARE

Motivated by the study in § 4 and § 5, we design and implement a proof-of-concept middleware called ODR (Offline Downloading Redirector) to help users get rid of performance bottlenecks. We evaluate the performance of ODR using real-world workloads.

6.1 Design and Implementation

ODR is presented as a public web service to users available at *http://odr.thucloud.com*. It is implemented as a middleware independent of any specific cloud-based offline downloading system or smart AP, and thus can be deployed on any dedicated servers or virtual machines. ODR takes users' offline downloading requests, and adaptively decides in which way the requested files should be downloaded to achieve the best expected downloading experience.

Specifically, when a user wants to download a file from the Internet, she first accesses the web service of ODR (by opening the front web page) and inputs the HTTP/FTP/P2P link to the original data source. In addition, ODR asks for other auxiliary information including the user's IP address, access bandwidth, smart AP type, storage device and filesystem type [4]. All the aforementioned information is straightforward for common users, except the access

[4]ODR maintains a web cookie at the user side (if her web browser permits), so that the user does not need to repeatedly input the auxiliary information every time.

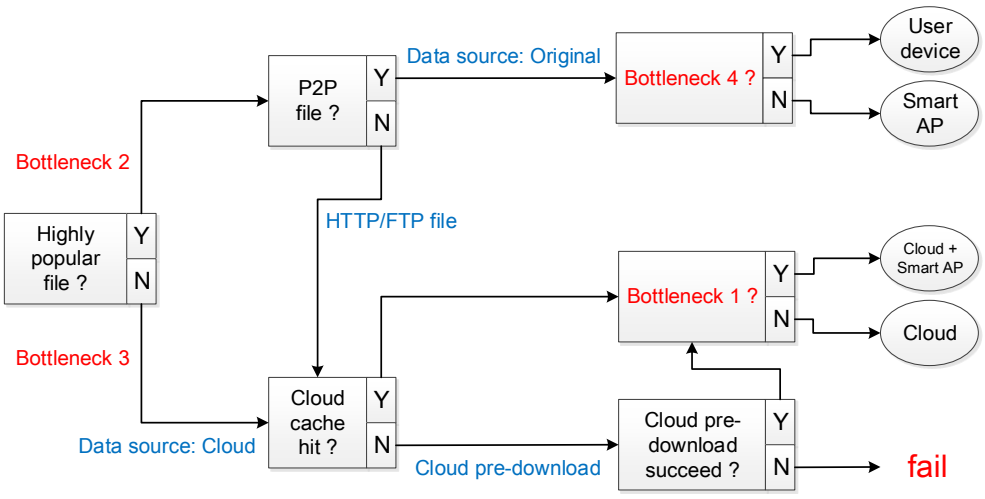

Figure 15: Workflow state transition diagram of ODR. "Y": Yes; "N": No. Bottleneck 1, Bottleneck 2, Bottleneck 3 and Bottleneck 4 denote the four performance bottlenecks of offline downloading mentioned in our key results (refer to § 1).

bandwidth. Fortunately, as the majority of China's Internet users have installed PC-assistant software such as Tencent PC Manager [37], Baidu Guard [11], and 360 Security Guard [2]. With simple instructions, ODR is able to guide users how to obtain the approximate value of access bandwidth with the PC-assistant software.

On receiving an offline downloading request, ODR firstly queries the content database to obtain the latest popularity statistics of the requested file. We use the Xuanfeng database in our implementation, while keep in mind that the performance of ODR would be further enhanced if it is able to use multiple cloud services (*e.g.,* Xuanfeng + Xunlei + Baidu CloudDisk) at once. ODR calculates the decisions based on the popularity of the file and the auxiliary information provided by the user. The decision is then returned to users via the front web page of the ODR service.

Figure 15 plots the state transition diagram of ODR's decision making process, which involves a series of conditions and branches as follows.

Handling highly popular files. First and foremost, users are concerned with the downloading success of a requested file. As the downloading failure ratio is tightly related with the popularity (refer to § 4.1 and § 5.2), ODR needs to examine whether the requested file is highly popular. If yes, the ODR is likely to be successful at downloading the file, and therefore we can make efforts to mitigate the cloud-side upload bandwidth burden (addressing Bottleneck 2 in Figure 15).

To deal with Bottleneck 2, if the highly popular file is hosted in a P2P data swarm, ODR suggests the user to directly download the file from its original data source (*i.e.,* the abundant peers sharing the file). On the contrary, if the highly popular file is hosted in an HTTP/FTP server, ODR would suggest the user to fall back on the cloud, so as to avoid making the HTTP/FTP server a bottleneck.

Further, to minimize the expected (pre-)downloading time, ODR considers the user-side access bandwidth, as well as the smart AP's storage device and filesystem type (addressing Bottleneck 4 in Figure 15). For example, when the user-side access bandwidth reaches 20 Mbps (= 2.5 MBps), if the smart AP uses a USB flash drive as its storage device or its storage device is formatted as NTFS, ODR would suggest the user to directly download the file using her local device (given that it is usually inconvenient for the user to change the storage device and filesystem of a smart AP during pre-downloading). On the other hand, when the user-side access bandwidth is below 0.93 MBps (refer to Table 2), ODR would suggest the user to utilize their smart AP.

Handling less popular files. When a requested file is not highly popular, the success of downloading is our primary concern (addressing Bottleneck 3 in Figure 15). To this end, ODR leverages the cloud storage pool to minimize the failure ratio of (pre-)downloading. Specifically, the downloading falls into the following two cases:

- *Case 1:* If the requested file is already cached in the cloud, ODR should further detect whether there is a bandwidth bottleneck between the cloud and the user by analyzing the user-side access bandwidth and ISP information [5] (addressing Bottleneck 1 in Figure 15). If the user-side access bandwidth is low ($<$ 1 Mbps = 125 KBps) or the user is located in a different ISP other than the four ISPs supported by the cloud, ODR would suggest the user to leverage both the cloud and their smart AP to mitigate the impediments of Bottleneck 1 ("Cloud + Smart AP" in Figure 15, *i.e.,* the file should be first pre-downloaded by the smart AP from the cloud, and then fetched by the user from the smart AP). Otherwise, ODR suggests the user to fetch from the cloud.

- *Case 2:* If the requested file is not cached in the cloud, ODR suggests the user to first use the cloud for pre-downloading. After the file is successfully pre-downlo- aded by the cloud, the user will be notified, and then she can ask ODR again for further suggestions (either directly fetching from the cloud, or from the cloud to a smart AP and then to her local device). If the cloud fails to download the file, the user will be notified of a pre-downloading failure.

Note that ODR never delivers file contents by itself, which makes its operation lightweight in terms of bandwidth and traffic consumption. For our implementation, we rent a low-end virtual machine from Aliyun.com (a major cloud service provider in China) to host the entire ODR service. This virtual machine has a public

[5]The user's ISP information is obtained based on her IP address with the help of the APNIC service (*http://www.apnic.net*), a major service provider for IP address collecting/resolving in Asia Pacific.

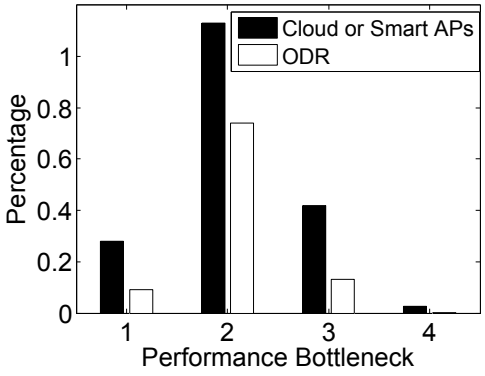

Figure 16: Benchmark performance of ODR, compared with Cloud (*i.e.,* the cloud-based approach) or Smart APs (*i.e.,* the smart AP based approach). Note that the Y-label of Bottleneck 2 is defined as $\frac{\text{Peak cloud bandwidth burden}}{\text{Purchased cloud bandwidth (30 Gbps)}}$.

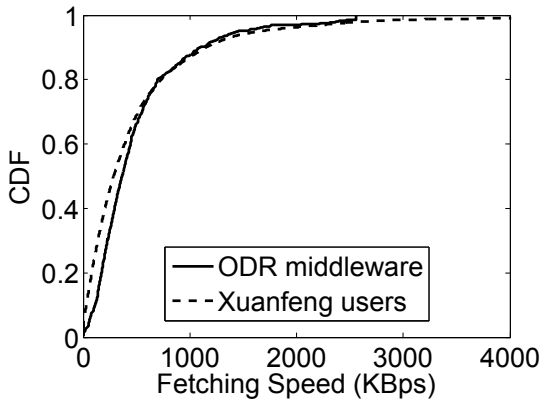

Figure 17: CDF of fetching speeds using ODR. Min: 0 KBps; Median: 368 KBps; Average: 509 KBps; Max: 2.37 MBps ($\approx$ 20 Mbps). As a comparison, the CDF of cloud-based fetching speeds is also plotted.

IP address and 1 Mbps (= 125 KBps) of Internet access bandwidth. The monthly operation cost of ODR is merely $20.

Limitation. By proposing the ODR middleware, our major goal is to demonstrate the potential benefits for a hybrid approach that can effectively address the limitations of the two existing conventional approaches while inheriting their respective advantages. Therefore, the above design and implementation of ODR is basically a *coarse-grained* solution to optimize the offline downloading performance and overhead, where the optimization granularity is a whole offline downloading request/task.

ODR can further benefit from more fine-grained alternative optimization solutions at the chunk, TCP data flow, or sub-stream levels. For instance, a more dynamic solution proposed by Huang *et al.* [58] (*i.e.,* a buffer-based adaptive bit rate selection algorithm for video streaming) can be used instead of the current hard coded decision procedure of ODR. Moreover, a simple solution (called "mobile phone content pre-staging" [52]) that has been proposed in the past is to simply defer downloads to later times when the download bandwidth is better if the users are not particularly time sensitive. In addition, there are even standards efforts like Low Extra Delay Background Transport (LEDBAT, IETF RFC 6817 [27]) that seeks to utilize the available bandwidth on an end-to-end path while limiting the resulting penalty of delay increase on that path. ODR can learn from LEDBAT to further mitigate the cloud-side upload bandwidth burden.

6.2 Performance Evaluation

To evaluate the performance of ODR, we replay the sampled workload (refer to § 5.1) using the deployed ODR middleware during Mar. 23–Apr. 13, 2015. The environment is similar to that in Figure 12. For each smart AP, we use a common laptop (with Quad-core Intel i5 CPU @1.70 GHz, 4-GB RAM, and 7200-RPM 500-GB hard disk drive) as the user device. Figure 16 shows the overall performance results of ODR, compared with the performances of Xuanfeng and the three popular smart APs.

First, the percentage of impeded fetching processes (Bottleneck 1) is reduced from 28% to 9%. The remainder (9%) is mostly due to the intrinsic dynamics of the Internet. In detail, Figure 17 shows the fetching speed distribution using ODR. Compared with the fetching speed distribution of Xuanfeng, the median fetching speed is enhanced from 287 KBps to 368 KBps. Limited by our benchmark environment, the max fetching speed using ODR is 20 Mbps (= 2.5 MBps), which is lower than that in Xuanfeng (*i.e.,* 50 Mbps = 6.25 MBps). This is why the average fetching speed using ODR (509 KBps) is comparable to that of Xuanfeng (504 KBps).

With regard to Bottleneck 2, the cloud-side upload bandwidth burden under the sampled workload is reduced by 35%, which is attributable to the fact that the cloud no longer needs to deliver highly popular P2P files. If Xuanfeng had integrated ODR, its peak upload bandwidth burden could decrease from 34 Gbps (refer to Figure 11) to around 22 Gbps, and thus Xuanfeng would not need to reject any fetching request, given their current workload.

Further, the percentage of smart APs' failures in pre-downloading unpopular files (Bottleneck 3) is reduced from 42% to 13%. In addition, Bottleneck 4 (caused by unsuitable storage devices or filesystems) is almost completely avoided with the use of ODR.

In a nutshell, the integration of ODR into offline downloading services results in a marked reduction of performance bottlenecks, as well as considerable improvement on the quality of service.

Limitation. We acknowledge that ODR occasionally makes incorrect redirection decisions due to the inherent dynamics of Internet that may degrade performance. Fortunately, the percentage of such decisions is negligible ($<$1%) in our evaluation. Thus, we believe that the performance of ODR is acceptable in practice, especially given the significant advantages of the system.

7. RELATED WORK

This section reviews previous studies of offline downloading, including both the cloud and smart AP based approaches. We also discuss the hybrid approach which directly connects the smart AP to the cloud and compare it with our ODR. In addition, we describe a few offline downloading (based) services outside China. Finally, we briefly survey the state-of-the-art downloading techniques and compare them with offline downloading.

Cloud-based approach. With the proliferation of cloud-based services, a number of studies have investigated the design and implementation of cloud-based offline downloading systems and their performance measurements.

Huang *et al.* [59] presented the early-stage system design of Xuanfeng in 2011, which focuses on guaranteeing data health and accelerating the downloading speed of unpopular videos. However, perhaps due to the startup stage of the system at that time (with

a relatively small user base), their study did not notice the two critical performance bottlenecks, *i.e.,* Bottleneck 1 and Bottleneck 2 uncovered in this paper.

Ao *et al.* [44] conducted a long-term measurement study of Xuanfeng in 2012, and predicted that the system would be short of cloud-side upload bandwidth in the near future. Complementary to their study, our work provides in-depth insights into the cloud-side upload bandwidth burden, and recognizes Bottleneck 2 — the root cause of cloud bandwidth shortage. Furthermore, our proposed ODR middleware can significantly reduce cloud-side bandwidth consumption by asking users to download highly popular P2P files directly from their data swarms.

Zhou *et al.* [72] made a theoretical analysis of offline downloading service models, including the cloud-based model and the peer-assisted model. They argue that the former can help scale with file population and the latter should be used to deal with popular files. An adaptive algorithm called AMS (Automatic Mode Selection) is proposed for selecting an appropriate model. Compared with AMS, ODR is more general and applicable: it requires no modification on the cloud side. Additionally, ODR is a deployed system rather than a theoretical algorithm.

Smart AP based approach. Since the history of smart APs is extremely short (just two years), we are not aware of any systematic study on their downloading performances. Most of the existing evaluations [6, 7, 8, 39, 23, 4] are based on simplified network environments and unrealistic workloads (*e.g.,* a few popular files). This is probably the reason why Bottleneck 3 in our study had not been discovered yet. Furthermore, Bottleneck 4 had never been identified in existing evaluations of smart APs.

Hybrid approach. HiWiFi, MiWiFi, and Newifi all provide hybrid solutions for offline downloading [19, 29, 32]. In these hybrid solutions, user-requested files are first downloaded by the cloud, and then the smart AP fetches the files from the cloud. That is to say, the downloading process always goes through the longest data flow: first from the Internet to the cloud, and then to the smart AP of the user.

In contrast, our ODR middleware adaptively selects the most efficient data flow for users to avoid performance bottlenecks. Therefore, ODR significantly outperforms the current hybrid approach by addressing the bottlenecks of both (cloud and smart AP based) approaches while also inheriting their advantages.

Offline downloading outside China. Besides those developing countries (as mentioned in § 1), developed countries can also benefit from offline downloading (based) services. For example, URL Droplet [40] and the Amazon Silk web browser [5] take advantage of the cloud to speed-up file transfers. The former employs the Dropbox cloud storage service to download and host files for its users; the latter utilizes Amazon's cloud servers to improve web page loading performance for Kindle Fire tablets and smartphones. On the other side, the smart AP based approach is adopted by the Linksys Smart WiFi Router [26] which is sold in the US.

State-of-the-art downloading techniques. Existing techniques for Internet file downloading mainly include the centralized Web-based (or client-server), the hierarchical CDN (*i.e.,* content delivery network like Akamai, L-3, and ChinaCache), the decentralized P2P, and the latest ICN (*i.e.,* information centric networking).

The Web-based technique is obviously subject to the intrinsic dynamics and heterogeneities of the Internet, particularly the single-point bottleneck of the Web server. By strategically deploying edge servers in numerous locations that are closer to users, the CDN technique effectively optimizes file downloading performance. Nevertheless, being a commercial service, CDN vendors typically only help to deliver files for *content providers* who pay for the service. On the contrary, the business model of offline downloading is the opposite of CDN, because it charges (or sometimes frees) its users, *i.e., content receivers*, for better downloading experiences.

As mentioned in § 3, the P2P technique is good at distributing popular files that are each shared by a number of users, but cannot guarantee the data availability or maintain a high speed for the downloads of unpopular files. This is why so many users have resorted to offline downloading for acquiring files that are originally hosted in P2P data swarms. In a word, offline downloading addresses the unstability and uncertainty of P2P under certain scenarios.

ICN, also known as CCN (content centric networking) or NDN (named data networking), is motivated by content receivers' interest in the network to achieve efficient and reliable data distribution [43], regardless of the properties of the original data sources. Specifically, ICN is featured by a number of desired functions, such as 1) in-network storage for caching, 2) decoupling content senders and receivers, 3) disruption tolerance, 4) multi-party communication through replication, 5) mobility and multi-homing, *etc*. It is easy to find that offline downloading fulfills at least the first three desired functions of ICN *in a real-world setting without breaking the current Internet architecture*. Further, we would like to explore whether and how offline downloading can enable the other functions of ICN.

8. CONCLUSION

In recent years, the Internet has gained enormous penetration all over the world. However, basic Internet services, like downloading (large) files, remains an issue in most developing countries as a consequence of low-quality network connections. To improve users' downloading experiences, offline downloading services have been proposed and widely deployed in China.

The idea of offline downloading is mainly embodied in two different types of approaches: (1) the cloud-based approach and (2) the smart AP based approach. Unfortunately, the two approaches are confusing to end users since they offer different strengths and weaknesses (the so-called "selection dilemma"). Our study addresses this dilemma with in-depth analysis of a large-scale cloud-based offline-downloading system, as well as comprehensive benchmark experiments of three popular smart APs. Our study shows that the two approaches are subject to distinct performance bottlenecks, while also being complementary to each other. Driven by the study, we build an ODR middleware to help users achieve the best expected performance.

In the future, we envision that offline downloading will become a widely used technology for enhancing the Internet experiences of users across both the developing world and the developed world. For example, people start to build a variety of useful Internet services on top of cloud-based offline downloading systems, such as cloud-based media converters (*e.g.,* Cloud Transcoder [65]) and cloud-accelerated web browsers (*e.g.,* QQ mobile web browser, UCWeb browser, and Amazon Silk web browser). Our study of offline downloading provides solid experiences and valuable heuristics for the developers of similar and relevant services.

9. ACKNOWLEDGEMENTS

This work is supported by the High-Tech Research and Development Program of China ("863 – China Cloud" Major Program) under grant 2015AA01A201, the China NSF under grant 61471217, the US NSF under grant CNS-1319019, the China Postdoctoral Sci-

ence Fund under grant 2014M550735, and the CCF-Tencent Open Fund under grant AGR20150201.

We would like to thank our shepherd Vyas Sekar and the Tencent Xuanfeng team for their valuable help.

10. REFERENCES

[1] 100,000 MiWiFi smart AP devices are sold out in just 59 seconds. http://bbs.xiaomi.cn/thread-9658495-1-1.html.

[2] 360 Security Guard. http://www.360.cn/weishi.

[3] A Collection of the Best Offline Downloading Tools. http://jingyan.baidu.com/article/636f38bb295e9bd6b84610e9.html.

[4] A Comparison of Nine Smart APs. http://net.zol.com.cn/478/4788494.html.

[5] Amazon Silk web browser. http://amazonsilk.wordpress.com.

[6] An Evaluation Report of HiWiFi. http://news.mydrivers.com/1/279/279305_all.htm.

[7] An Evaluation Report of MiWiFi. http://www.geekpark.net/read/view/195133.

[8] An Evaluation Report of NewiFi. http://www.itbear.com.cn/n112446c93.aspx.

[9] aria2: The next generation download utility. http://aria2.sourceforge.net.

[10] Baidu CloudDisk offline downloading system. http://pan.baidu.com.

[11] Baidu Guard. http://anquan.baidu.com/weishi.

[12] CERNET (China Education and Research Network) ISP. http://www.cernet.edu.cn.

[13] China-Mobile ISP. http://www.10086.cn.

[14] China-Telecom ISP. http://www.chinatelecom.com.cn.

[15] China-Unicom ISP. http://www.chinaunicom.com.cn.

[16] China's Broadband Penetration Is Increasingly Lagging Behind Developed Nations, Says MIIT's Research Head. http://techcrunch.com/2013/03/21/china-broadband-laggin.

[17] Delay-tolerant networking (DTN) Wiki page. http://en.wikipedia.org/wiki/Delay-tolerant_networking.

[18] FCC raises broadband definition to 25Mbps, Chairman mocks ISPs. http://www.extremetech.com/mobile/198583-fcc-raises-broadband-definition-to-25mbps-chairman-mocks-isps.

[19] HiWiFi APPs. http://bbs.hiwifi.com/thread-22663-1-1.html.

[20] HiWiFi Introduction and History. http://www.hiwifi.com/about.

[21] HiWiFi smart AP. http://www.hiwifi.com.

[22] HiWiFi smart AP is striving towards 5,000,000 sales. http://www.pcpop.com/doc/1/1002/1002782.shtml.

[23] HiWiFi vs. Newifi: A Benchmark Test. http://test.smzdm.com/pingce/p/20574.

[24] How to Offline Download? Which is the Best Offline Downloading Tool? http://jingyan.baidu.com/article/a65957f4fe63c424e67f9b92.html.

[25] Key ICT indicators for developed and developing countries and the world (totals and penetration rates). http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2014/ITU_Key_2005-2014_ICT_data.xls.

[26] Linksys Smart WiFi Router. http://www.linksys.com/en-us/smartwifi.

[27] Low Extra Delay Background Transport (LEDBAT), IETF RFC 6817. http://datatracker.ietf.org/doc/rfc6817.

[28] MiWiFi smart AP. http://www.miwifi.com.

[29] MiWiFi: System Options. http://www.mi.com/miwifi#op.

[30] Newifi smart AP. http://www.newifi.com.

[31] Offline Downloading: the Baidu Wikipedia page. http://baike.baidu.com/view/2718066.htm.

[32] Offline Downloading with Newifi and Xunlei. http://jingyan.baidu.com/article/3d69c5517049e1f0cf02d7a4.html.

[33] Offline (movie) Downloading for MiWiFi. http://www.mi.com/miwifi/movie-download.

[34] OpenWrt operating system. http://openwrt.org.

[35] Opkg (Open PacKaGe Management) web site. https://code.google.com/p/opkg.

[36] Statistics of China Internet Users. http://www.internetlivestats.com/internet-users/china.

[37] Tencent PC Manager. http://guanjia.qq.com.

[38] The State of Broadband 2014 – A Report by the Broadband Commission. http://www.broadbandcommission.org/documents/reports/bb-annualreport2014.pdf.

[39] Three Faults of HiWiFi. http://digi.tech.qq.com/a/20131112/002037.htm.

[40] URL Droplet offline downloading system. http://www.urldroplet.com.

[41] Xuanfeng offline downloading system. http://xf.qq.com.

[42] Xunlei offline downloading system. http://lixian.xunlei.com.

[43] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. A Survey of Information-Centric Networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[44] Naixiang Ao, Yingying Xu, Changjia Chen, and Yuchun Guo. Offline Downloading: A Non-Traditional Cloud-Accelerated and Peer-Assisted Content Distribution Service. In *Proc. of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 81–88. IEEE, 2012.

[45] Zachary S Bischof, Fabián E Bustamante, and Rade Stanojevic. Need, Want, Can Afford ÍC Broadband Markets and the Behavior of Users. In *Proc. of the 14th ACM Internet Measurement Conference (IMC)*, pages 73–86. ACM, 2014.

[46] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 126–134. IEEE, 1999.

[47] Guihai Chen and Zhenhua Li. *Peer-to-Peer Network: Structure, Application and Design*. Tsinghua University Press, 2007.

[48] Marshini Chetty, Srikanth Sundaresan, Sachit Muckaden, Nick Feamster, and Enrico Calandro. Measuring Broadband Performance in South Africa. In *Proc. of the 4th Annual Symposium on Computing for Development (DEV), Rondebosch Cape Town, South Africa*. ACM, 2013.

[49] David Choffnes and Fabián E Bustamante. Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems. *ACM SIGCOMM Computer Communication Review (CCR)*, 38(4):363–374, 2008.

[50] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems*, pages 68–72, 2003.

[51] Amal Fahad, Zhuan Chen, Kai Shen, Jeffrey Bigham, and Assmaa Fahad. An Evaluation of Web Acceleration Techniques for the Developing World. In *Proc. of the 6th USENIX/ACM Workshop on Networked Systems for Developing Regions (NSDR), Boston, MA*, 2012.

[52] Alessandro Finamore, Marco Mellia, Zafar Gilani, Konstantina Papagiannaki, Vijay Erramilli, and Yan Grunenberger. Is There a Case for Mobile Phone Content Pre-staging? In *Proc. of the 9th ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 321–326. ACM, 2013.

[53] Aditya Ganjam, Junchen Jiang, Xi Liu, Vyas Sekar, Faisal Siddiqi, Ion Stoica, Jibin Zhan, and Hui Zhang. C3: Internet-Scale Control Plane for Video Quality Optimization. In *Proc. of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2015.

[54] Sarthak Grover, Mi Seon Park, Srikanth Sundaresan, Sam Burnett, Hyojoon Kim, Bharath Ravi, and Nick Feamster. Peeking Behind the NAT: An Empirical Study of Home Networks. In *Proc. of the 13th ACM Internet Measurement Conference (IMC)*, pages 377–390. ACM, 2013.

[55] Krishna P Gummadi, Richard J Dunn, Stefan Saroiu, Steven D Gribble, Henry M Levy, and John Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-sharing Workload. *ACM SIGOPS Operating Systems Review (OSR)*, 37(5):314–329, 2003.

[56] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, and Xiaodong Zhang. The Stretched Exponential Distribution of Internet Media Access Patterns. In *Proc. of the 27th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 283–294. ACM, 2008.

[57] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proc. of the 12th ACM Internet Measurement Conference (IMC)*, pages 225–238. ACM, 2012.

[58] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proc. of the 2014 ACM Conference on Communication Architectures, Protocols and Applications (SIGCOMM)*, pages 187–198. ACM, 2014.

[59] Yan Huang, Zhenhua Li, Gang Liu, and Yafei Dai. Cloud Download: Using Cloud Utilities to Achieve High-quality Content Distribution for Unpopular Videos. In *Proc. of the 19th ACM International Conference on Multimedia (MM)*, pages 213–222. ACM, 2011.

[60] Sibren Isaacman and Margaret Martonosi. Low-Infrastructure Methods to Improve Internet Access for Mobile Users in Emerging Regions. In *Proc. of the 20th International Conference on World Wide Web (WWW)*, pages 473–482. ACM, 2011.

[61] David Johnson, Elizabeth Belding, and Consider Mudenda. Kwaabana: File Sharing for Rural Networks. In *Proc. of the 4th Annual Symposium on Computing for Development (DEV), Rondebosch Cape Town, South Africa*. ACM, 2013.

[62] Shunmuga Krishnan and Ramesh Sitaraman. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs. *IEEE/ACM Transactions on Networking (TON)*, 21(6):2001–2014, 2013.

[63] Zhenhua Li, Jiannong Cao, and Guihai Chen. ContinuStreaming: Achieving High Playback Continuity of Gossip-based Peer-to-Peer Streaming. In *Proc. of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–12. IEEE, 2008.

[64] Zhenhua Li, Yan Huang, Gang Liu, Fuchen Wang, Yunhao Liu, Zhi-Li Zhang, and Yafei Dai. Challenges, Designs, and Performances of Large-Scale Open-P2SP Content Distribution. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 24(11):2181–2191, 2013.

[65] Zhenhua Li, Yan Huang, Gang Liu, Fuchen Wang, Zhi-Li Zhang, and Yafei Dai. Cloud Transcoder: Bridging the Format and Resolution Gap between Internet Videos and Mobile Devices. In *Proc. of the 22nd SIGMM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 33–38. ACM, 2012.

[66] Zhenhua Li, Tieying Zhang, Yan Huang, Zhi-Li Zhang, and Yafei Dai. Maximizing the Bandwidth Multiplier Effect for Hybrid Cloud-P2P Content Distribution. In *Proc. of the 20th IEEE/ACM International Workshop on Quality of Service (IWQoS)*, pages 1–9. IEEE, 2012.

[67] Srikanth Sundaresan, Walter De Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Broadband Internet Performance: A View From the Gateway. *ACM SIGCOMM Computer Communication Review (CCR)*, 41(4):134–145, 2011.

[68] Ye Tian, Ratan Dey, Yong Liu, and Keith Ross. Topology Mapping and Geolocating for China's Internet. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 24(9):1908–1917, 2013.

[69] Haiyong Xie, Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. P4P: Provider Portal for Applications. *ACM SIGCOMM Computer Communication Review (CCR)*, 38(4):351–362, 2008.

[70] Yasir Zaki, Jay Chen, Thomas Pötsch, Talal Ahmad, and Lakshminarayanan Subramanian. Dissecting Web Latency in Ghana. In *Proc. of the 14th ACM Internet Measurement Conference (IMC)*, pages 241–248. ACM, 2014.

[71] Mariya Zheleva, Paul Schmitt, Morgan Vigil, and Elizabeth Belding. The Increased Bandwidth Fallacy: Performance and Usage in Rural Zambia. In *Proc. of the 4th Annual Symposium on Computing for Development (DEV), Rondebosch Cape Town, South Africa*. ACM, 2013.

[72] Yipeng Zhou, Zhengjia Fu, Dah-Ming Chiu, and Yan Huang. An Adaptive Cloud Downloading Service. *IEEE Trans. on Multimedia (TMM)*, 15(4):802–810, 2013.