Design, Implementation, and Measurement of a Crowdsourcing-Based Content Distribution Platform

YIPENG ZHOU and LIANG CHEN, Shenzhen University
MI JING and SHENGLONG ZOU, Xunlei Networking Technologies Ltd.
RICHARD TIANBAI MA, National University of Singapore

Content distribution, especially the distribution of video content, unavoidably consumes bandwidth resources heavily. Internet content providers invest heavily in purchasing content distribution network (CDN) services. By deploying tens of thousands of edge servers close to end users, CDN companies are able to distribute content efficiently and effectively, but at considerable cost. Thus, it is of great importance to develop a new system that distributes content at a lower cost but comparable service quality. In lieu of expensive CDN systems, we implement a crowdsourcing-based content distribution system, Thunder Crystal, by renting bandwidth for content upload/download and storage for content cache from agents. This is a large-scale system with tens of thousands of agents, whose resources significantly amplify Thunder Crystal's content distribution capacity. The involved agents are either from ordinary Internet users or enterprises. Monetary rewards are paid to agents based on their upload traffic so as to motivate them to keep contributing resources. As far as we know, this is a novel system that has not been studied or implemented before. This article introduces the design principles and implementation details before presenting the measurement study. In summary, with the help of agent devices, Thunder Crystal is able to reduce the content distribution cost by one half and amplify the content distribution capacity by 11 to 15 times.

CCS Concepts: • **Information systems → Multimedia streaming; Incentive schemes;** • **Networks** → *Peer-to-peer networks;*

Additional Key Words and Phrases: Crowdsourcing, video distribution, agent, CDN

ACM Reference Format:
Yipeng Zhou, Liang Chen, Mi Jing, Shenglong Zou, and Richard Tianbai Ma. 2016. Design, implementation, and measurement of a crowdsourcing-based content distribution platform. ACM Trans. Multimedia Comput. Commun. Appl. 12, 5s, Article 80 (November 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2978655

1. INTRODUCTION

Video content is regarded as the future of the Internet content market. It is anticipated by Cisco [2012] that Internet traffic taken by video content will increase from 64% in 2014 to 84% in 2019. There exist many reputed global Internet content providers

The research was partially supported by the Foundation of Shenzhen City (no. KQCX20140519103756206), the Natural Science Foundation of China (nos. 61402297 and 61502315), the Natural Science Foundation of Guangdong province (no. 2015A030310366), and the Shenzhen Science and Technology Foundation (grant JCYJ20150529164656096).
Authors' addresses: Y. Zhou, Room 622-3, College of Computer Science and Software Engineering, Shenzhen University, Nanhai Avenue 3688, Shenzhen 518060, P.R.China; email: ypzhou@szu.edu.cn; L. Chen (corresponding author), N910, College of Information Engineering, Shenzhen University, Nanhai Avenue 3688, Shenzhen 518060, P.R.China; email: lchen@szu.edu.cn; M. Jing and S. Zou, Floor 8, Xunlei Networking Technologies Ltd., Software Park 2, Keji Mid 2nd Road, Nanshan District, Shenzhen 518060, P.R.China; emails: {jingmi, sean}@xunlei.com; R. T. Ma, #04-27, 15 Computing Drive, COM2 Building, School of Computing, National University of Singapore, S(117418); email: tbma@comp.nus.edu.sg.

DOI: http://dx.doi.org/10.1145/2978655

that deliver various video content to billions of users, such as YouTube,[1] Netflix,[2] and Tencent Video.[3] For content providers, it is challenging to distribute videos to a great number of users via the Internet. Although network capacity has been expanded greatly over the past several years, it still cannot accommodate the rapidly increasing demand for video content. In particular, the demands for (ultra) high definition (HD) and 3D videos will make the problem even more challenging. Thus, it is of great value to develop novel content distribution platforms. In this study, we introduce a novel crowdsourcing-based content distribution platform, Thunder Crystal, which can efficiently distribute content at low cost. In principle, Thunder Crystal is a platform that can distribute any kind of content. However, this study focuses on the most challenging video distribution case because video distribution is the application with the most heavy bandwidth consumption and stringent quality requirement [Cha et al. 2007]. For convenience, we use the term *content* interchangeably with the term *video* in this article.

The content distribution network (CDN) and peer-to-peer (P2P) network are two common solutions adopted by content providers for video distribution [Passarella 2012]. CDN servers are maintained and operated by commercial companies that provide professional service to accelerate content delivery. A widely used CDN architecture deploys a large number of edge servers close to end users to reduce the transmission delay [Adhikari et al. 2012]. A sufficient number of deployed edge servers are also needed to guarantee service quality. Each edge server must be equipped with storage to cache the most popular content, although the storage of a single edge server is rather limited so that only a small fraction of videos can be cached. The hit rate (i.e., the probability that the requested content sent by a user is cached by the closest edge server) is one of the important metrics to measure CDN performance. If the requested content is not found in the server (i.e., missed), the user request has to be redirected to remoter servers to download the content, possibly resulting in longer transmission latency [Adhikari et al. 2012]. The replication algorithm placing appropriate content on a set of edge servers is one of the key design issues in CDN systems. In addition, edge servers update the cached content in a timely fashion by pulling the requested missing video content from central servers since the interests of Internet users change very fast. Thus, the maintenance and operation of a CDN system is quite expensive, implying that content providers have to support an expensive CDN service. In Thunder Crystal, a similar content replication strategy must be designed to update the content on agent devices, and the hit rate is adopted as the metric to measure the performance.

In contrast, P2P networks organize users to download the same content to communicate and share content with each other by forming overlay networks [Cohen 2003]. By freely using users' resources, P2P is a cheap method of content distribution. In addition, the capacity to distribute a file by P2P systems will automatically scale up with the user population exchanging the particular file. However, the P2P system has several flaws that restrict its applications: (1) users' quality of experience (QoE) could be impaired by frequent disk I/O, as users need to upload and download content simultaneously; (2) as a large-scale distributed system, it is difficult to protect content copyright because the shared content between users could be abused; and (3) P2P cannot provide high-quality services due to users' unstable network resources and highly dynamic behavior. For example, for an unpopular video, it may be difficult for users to share content because of user scarcity. Due to the preceding weaknesses, P2P cannot support video distribution very well independently. Instead, hybrid CDN/P2P is widely

[1]https://www.youtube.com.
[2]https://www.netflix.com.
[3]http://v.qq.com/.

adopted in real systems, in which the P2P system is implemented as a complementary system to CDN to reduce distribution cost as much as possible.

Different from the preceding two systems, we design and implement Thunder Crystal, a novel crowdsourcing-based content distribution platform. Our goal is to build a system providing service quality comparable to CDNs but with much lower costs. Briefly speaking, the design principle is to build the system by renting storage and bandwidth resources from users or enterprises, regarded as *agents*, instead of deploying many edge servers. To motivate agents to contribute resources, monetary rewards are paid to each agent based on his upload traffic. Agents do not control the content caching or upload and download strategies, except for setting the upper limits for the contributed bandwidth and storage resources. More intuitively, agent devices can be operated as mini-CDN servers under Thunder Crystal's control. To maintain the intrinsic attractiveness of the content cached on agent devices, Thunder servers continuously push the latest video content to each agent. Interestingly, to cache more content to earn more crystals, agents with more available bandwidth resources would desire to contribute more storage resources. Compared to CDNs, the equipment investment of Thunder Crystal is much lower, as the expenses to deploy data centers and edge servers are saved by renting resources from agents. Compared to the P2P system, Thunder Crystal provides more reliable service because of active content replication and more stable agent availability. Content copyright is protected by encrypting the content before it is pushed to agent devices. Taking all factors into account, including cost, service quality, and copyright protection, we find that Thunder Crystal is an attractive and practical solution for content distribution.

This work presents the design, implementation, and measurement study of Thunder Crystal. More specifically, we introduce the design principles and implementation details of Thunder Crystal, and then conduct a measurement study, measuring agent behavior, resource distributions contributed by agents, and system performance. We also design performance metrics to evaluate both agent efficiency and whole system efficiency. Open research issues, such as the optimal content caching strategy and agent incentive mechanism, are discussed as future work to further enhance system performance. To the best of our knowledge, this is the first work toward building a real crowdsourcing-based content distribution platform.

The rest of the article is organized as follows. The background of agents and agent devices are introduced in Section 2. The design principles, system architecture, and function components are discussed in Section 3. Measurement results are presented in Section 4, and open research issues are discussed in Section 5. The contribution of our work in light of related works is discussed in Section 6. We conclude the article in Section 7.

2. AGENT AND AGENT DEVICE

In Thunder Crystal, the basic unit that undertakes content distribution is an agent who uses his devices. Each agent device can be considered as an atom of the Thunder Crystal system, which is composed of tens of thousands of agent devices. In this section, we briefly introduce the background of agents, how agents are involved in the Thunder Crystal system, and the devices used by them.

The agents in Thunder Crystal are ordinary Internet users or enterprises who have surplus bandwidth resources. It is well known that both users and enterprises as the clients of Internet service providers (ISPs) purchase Internet access service plans, providing certain upload/download capacity. However, almost no Internet users or enterprises can always fully utilize the bandwidth resources by saturating the uploading rate toward upload capacity. More or less, ISP clients have surplus bandwidth resources that can be purchased for a lower price for content distribution by Thunder Crystal.

Thus, any user or enterprise can apply to join Thunder Crystal as an agent. Normally, each agent can use at most 5 devices (that could be set up in different locations) for content distribution. However, for some powerful agents, each can contribute more than 30 devices. The former agent type is called a *thin agent*, whereas the latter is called a *fat agent*.

Monetary rewards based on upload traffic contributed to Thunder Crystal are paid each month as an incentive to agents. The traffic uploaded by each agent is converted into crystal, the currency used by the system for accounting purposes, with different rates. During peak hours from 12:00 to 22:00, 1GB traffic is equivalent to 1 crystal; otherwise, it is only 0.5 crystals. The payment for each crystal is 0.07 RMB, or about 0.011 USD. Although the payment for each crystal is much lower than the bandwidth cost paid for ISPs by CDN companies, agents have been sufficiently motivated by the bandwidth purchasing scheme. In addition to the bandwidth resources, agents also need to contribute storage resources to cache videos, but they will not be counted for monetary reward. To allow flexibility, agents can set up the limits of the upload rate and the storage size contributed to Thunder Crystal freely.

In Thunder Crystal, about half of the agents use personal computers (PCs) and the remaining agents use smart (Internet) access points (APs) as the devices for content caching and uploading. Smart APs are a kind of emerging device providing Internet access service (with wireless AP function). They are equipped with embedded operating systems and large storage, typically around 1TB, so that they can complete some tasks independently. Compared to PCs, smart APs have smaller size, much lower price, and lower power consumption. The electronic power cost is one of the major costs for agents working in Thunder Crystal, which could significantly reduce agents' profit margin and impair the agent incentive. Thus, with much lower energy consumption, the smart AP is a more attractive device for agents. We expect that more agents will adopt smart APs in the future, although they are not forced to do so. The smart AP is becoming popular, and it can be found on Taobao[4] and JD,[5] which are the most popular e-commerce systems in China.

The rest of the article takes each agent device as the unit for study. Although our discussion will differentiate fat agent devices and thin agent devices explicitly, Thunder Crystal treats them uniformly for monetary reward.

3. SYSTEM DESIGN

In this section, we introduce the system design principles. The big picture of the system architecture will be presented before we zoom into the discussion of the functionality of each component. The details of strategies playing key roles to determine system performance will be introduced later.

3.1. System Architecture and Key Components

Recall that Thunder Crystal accomplishes content distribution for content providers by renting resources from agents. However, agents will be in an idle state if they do not have the content requested by consumers. Thus, a cluster of Thunder servers needs to push (possibly the latest popular) content handed over by content providers to each agent. At the same time, software running on each agent device is needed to support content caching, content uploading, information reporting, and so on. The following is a more detailed introduction of the Thunder servers and functionalities of the software running on agents:

[4]https://world.taobao.com.
[5]http://www.jd.com.

—*Thunder servers* are in charge of information collection, user request redirection, and content pushing. First, Thunder servers need to collect pertinent information, including but not limited to the cached content, available storage, estimated upload capacity, accumulated crystals, and the system load for each agent. The data used in this study are sampled from the data collected through Thunder servers. Second, the users requesting any content will send requests to content providers, which will redirect user requests to Thunder Crystal. Based on the knowledge of cached content and system load on each agent device, the Thunder servers will redirect user requests again to those light loaded agent devices caching requested content. Third, it is not difficult to understand that content, especially video content, loses users' eyeballs very quickly. Thanks to the advance in content production technology, the content generation speed is much higher than ever before. Thus, it is necessary to continuously update the content cached on agents through pushing the latest content from Thunder servers. In our implemented system, 80TB traffic will be consumed per day to update content, which is the primary cost of Thunder Crystal.

—The *software* running on the agent devices reports necessary information to Thunder servers, manages the cached content, and serves user requests. It is also the interface through which agents set the bandwidth and storage limits contributed to Thunder Crystal. First, the number of crystals accumulated by an agent will be counted based on the upload traffic of all of his devices, before the information is reported to Thunder Crystal by the software. Second, the software is in charge of managing storage and reserving content copyright. Any abnormal event will be reported immediately in case of content abuse. Third, each agent device serves the user requests, assigned by Thunder Crystal, with best efforts. The system load of the agent device will be reported to Thunder Crystal periodically in case any agent device is overloaded by getting assigned too many user requests.

—*Content providers* provide content in various forms for Internet users. Their business mainly centers on video content buying, user quality of service (QoS), user QoE, video content recommendation, and so on. In most cases, instead of building expensive data centers and edge servers everywhere, content providers resort to CDNs for professional content distribution. The service of Thunder Crystal is a complementary service of traditional CDN service. However, by renting resources from agents, infrastructure building cost is reduced dramatically, which enables Thunder Crystal to provide service for content providers at a much lower price than traditional CDN companies.

Figure 1 shows the big picture of Thunder Crystal. It is a complicated system with three major components: Thunder servers, agents, and users. These components, interacting with each other, cannot work very well unless we design a series of strategies for them. Each strategy needs to consider how to optimize or balance several different goals, requiring careful design. For brevity, we summarize the design details of the most important three strategies used in Thunder Crystal as follows:

—The *content replication strategy* determines how the content is placed on each agent device. It must adapt to heterogeneous storage sizes while maintaining simplicity for content management and video content copyright protection. The underlying reasons to achieve these goals are explained as follows. According to our measurement, the storage size contributed by each agent device varies in a broad range from about 10GB to more than 20TB, whereas the size of most video files varies from 100MB to 10GB. Thus, files must be cut and reorganized such that they can be uniformly pushed to any agent device. To serve user requests, Thunder Crystal needs to know the global information of the content placed on each agent device. A simple content management scheme is required so that Thunder servers can get this information

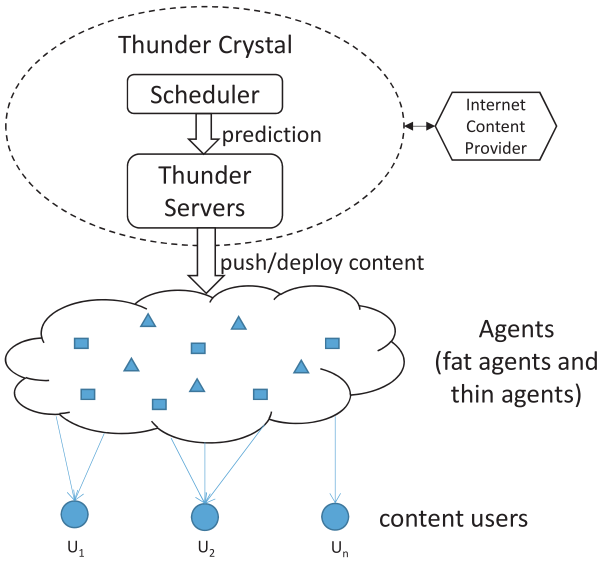

Fig. 1. System overview and general architecture.

at a low cost. Finally, in most cases, content providers request to preserve content copyright. The replication strategy needs to encrypt content to stop agents from abusing content.

—The *content pushing strategy* determines what content will be pushed to which agent device. The content, especially video content (e.g., movie, TV, and news) loses users' interest very quickly. The new video content keeps arriving at a certain rate. To fully utilize the bandwidth resources of agent devices, Thunder servers have to continuously push the latest video content to agent devices. The challenge is how to allocate the limited pushing bandwidth resources among different files.

—The *request scheduling and chunk transmission strategy* defines how agent devices serve user requests. Users access content through the interfaces provided by content providers before the user requests are redirected to Thunder Crystal and served by a set of agent devices caching the requested content. It is a many-to-many relationship between user requests and agent devices. The request scheduling and chunk transmission strategy specifies how to schedule multiple agent devices to serve a user request, and how to serve multiple user requests by a single-agent device simultaneously.

3.2. Business Model

The business model of Thunder Crystal is quite similar to CDN services with the exception that it rents resources from agents. Thunder Crystal provides general content distribution service, including but not limited to videos, documents, software, and games, for content providers (e.g., Tencent Video) to generate revenue. Based on the service orders made by content providers, Thunder Crystal recruits agents to match the bandwidth demands of content providers. In the current version, according to content providers' needs, about 11K active agent devices are involved that can provide sufficient bandwidth resources. For commercial CDN services, 1GB traffic may be used as the pricing unit, which is also adopted by Thunder Crystal. The price, 0.07 RMB for a 1GB upload, is attractive enough to recruit more than 11K active agent devices. In fact, 0.07 RMB is a pretty low price to ensure that Thunder Crystal can make a profit. In

Table I. Summary of Content Units and Their Purposes

	Designed For	Size
Video	Entire video file	$>$1GB
Chunk	Unit for encryption and indexing	300MB
Segment	Unit for storage in agent	512MB
Block	Unit for transmission	$<$2MB

contrast, the price of 1GB traffic uploaded by commercial CDNs is 0.26 RMB[6] in China. Thunder Crystal charges about 0.13 RMB for 1GB traffic, half of the commercial CDN price, as the CDN service has greater reliability and better performance. To keep agent devices active in uploading, it is necessary to push the latest video content to them. Based on the update rate of content and device population, roughly an 80TB server bandwidth is required.

Video downloading, especially HD video downloading,[7] has been dominating the content distribution service provided by Thunder Crystal. For those HD videos,[8] the size is about 1GB per file,[9] which has been taken into account by the design of Thunder Crystal. We split the files into chunks, segments, and blocks for the purpose of realizing the preceding three strategies. We summarize the content units and their design purposes in Table I, which will be further discussed in the next section with detailed design principles for each strategy. The parameter values shown in Table I are determined with the goal to optimize the distribution of large files with size around 1GB.

3.3. Important Strategies

The strategies introduced in this section are essential to system efficiency. Although these strategies are discussed one by one, they interact with each other. How to quantify the interactions and how to evaluate the performance of strategies are still open questions. We leave the discussion of the open research problems encountered by system design and implementation in Section 5.

3.3.1. Content Replication Strategy. We want the replicated content on agent devices to be transparent for agents. On the one hand, replicating a complete unencrypted file on agents' devices is vulnerable to copyright infringement. Agents could abuse cached content at will. The content could be redelivered to others by agents without notifying the content owners. On the other hand, agents could easily generate fake downloads if the replicated content is known. For example, an agent could be a home user who also downloads content for personal consumption. If the cached content is known, the agent could generate an infinite number of requests to download the content cached by his friends or himself to earn infinite crystals. This is similar to the coalitions game played by peers who know each other in private P2P networks, who cheat the reputation system by reporting fake download information to earn credits [Liu et al. 2010a]. Actually, this fake behavior can be prevented if the cached content is unknown to agents in Thunder Crystal.

In Thunder Crystal, each file is split into multiple chunks of 300MB each. The size of most files is about 1GB. Therefore, the overhead will be significant if the chunk size is too small. Chunks are encrypted before they are pushed out to different agents. Each file is split into at least two chunks in case an agent owns the complete file. For

[6]https://www.aliyun.com/.
[7]Actually, the video streaming is more prevalent. However, the current version can only support downloading. Supporting video streaming is our future work.
[8]Most HD videos belong to movie and television video (TV).
[9]In the rest of the article, we use *file* and *video* interchangeably.

example, a 500MB file will be split into two chunks: one with 300MB and the other with 200MB. However, if a video file is 220MB, it is still cut into two chunks of 110MB each. On agent devices, the chunks are then reorganized and assembled into segments of 512MB for each segment so that more than one chunk is contained in each segment, which increases the difficulty in recovering the cached content by agents. For agents, only the segments are seeable. In other words, there are two indices: segment index and chunk index. For agents, they can only find segments in their devices. For Thunder Crystal, the user requests are directed to the agents with target chunks based on the chunk index.

3.3.2. Content Pushing Strategy. To maintain system efficiency, Thunder servers need to push out the latest video content continuously to each device to ensure that agents have available content to upload. Our measurement finds that content popularity declines very fast, as content consumers' attention mainly fixes on newly created video content. The content pushing strategy determines what video files should be pushed to which agent device, which is a rather complicated problem. The optimal content pushing strategy is still unknown, and searching it could be an NP-complete problem. Instead, we turn to using a heuristic algorithm to determine the number of pushed copies for each file.

The bandwidth of Thunder servers that can be used for content pushing is limited. Thus, we need a strategy to set the number of pushed replicas for each file. For file j, suppose that if the number of requests to download file j is N_j on the previous day, then we will maintain $M_j = (\beta N_j)^\alpha$ replicas for file j. Here, α and β are set to be 0.96 and 0.05, respectively. The calculation of M_j is an empirical equation obtained by experimental trials. If there already exist m_j replicas in the system, $M_j - m_j$ replicas will be pushed out. However, it is not guaranteed that the push requirement of all files can be satisfied. The capacity of Thunder servers is only able to push 80TB traffic per day, as the traffic budget is limited. Actually, the most costly step in Thunder Crystal is the pushing of content from servers to agent devices. Thus, with limited resources, files will be pushed in decreasing order of popularity (i.e., decreasing order of N_j) until the 80TB traffic per day is exhausted. In other words, the deployment requirement of more popular files will be satisfied with higher priority. If an agent's storage is fully occupied, the files with $m_j > M_j$ will be removed to empty the space for caching new content.

In the content pushing strategy, agent devices are not discriminated actively. Thunder servers just push the new replicas randomly to available agent devices without any file-pushing task. Thus, the devices with a faster downloading rate tend to update content more quickly. Designing a more sophisticated pushing strategy and refining the empirical calculation of M_j will be our future work.

The content pushing strategy is still applicable for two special cases: newly offered files and files with flash crowds requests. For the former case, the initial popularity of the new files on the first day is estimated with heuristic algorithms by content providers who purchased services from Thunder Crystal. For the latter case, there are very few files with flash-crowded requests in Thunder Crystal, as most files are HD movies or TV, whose popularity is quite stable on a daily basis. In addition, it is likely to avoid the impairment caused by a flash crowd by overestimating the initial popularity for new videos, as a flash crowd most likely occurs for new files.

3.3.3. Request Scheduling and Chunk Transmission. Now we introduce the last strategy that schedules user requests to the right agent devices to fetch requested chunks. Each agent device serves the received user requests one by one with best efforts. For each user, the file downloading is executed chunk by chunk. A user request is split into multiple chunk requests. The request of the first chunk is sent to Thunder servers before it is redirected to the right agent devices. After completing the download of the first chunk, the request for the second chunk will be issued. For each chunk request, addresses of 200 agent devices chosen randomly from all devices replicating the requested chunk will be returned to the user. The user will contact these agent devices to download content simultaneously.

Once the connection between the user and the agent device is created by request scheduling strategy, the file content will be delivered block by block. Each chunk is divided into multiple exclusive blocks at a size of 2MB for each block. However, some chunks may not be divisible by 2MB. For some chunks with a size less than 300MB, it is possible that the tail part is less than 2MB. If the tail is larger than 512KB, the tail part will be taken as a single block; otherwise, dummy content will be supplemented to assemble a 512KB block (for transmission convenience). Using blocks as the transmission unit, users can download an individual chunk simultaneously from multiple agent devices. A scheduler is in charge of scheduling of blocks to avoid downloading duplicate blocks.

3.4. Merits of Thunder Crystal

The architecture of Thunder Crystal seems similar to the architectures of P2P networks; however, the major difference is the incentive mechanism to organize agents to contribute resources for content caching and distribution. In Thunder Crystal, agents will receive cash rewards based on their traffic used to upload content.

First, the incentive mechanism in Thunder Crystal is more effective in motivating agents without producing negative effects. In public P2P networks, to motivate peers uploading content as fast as possible [Cohen 2003; Fan et al. 2006], peers' downloading performance is mainly determined by their content uploading rates. However, peers merely upload the content being downloaded and usually refuse to cooperate once their file downloading is completed. In private P2P networks, a long-term reputation will be maintained for each peer based on his uploading and downloading of traffic since the user's registration is on the network. This mechanism is effective in motivating peers to keep uploading even if the downloading is finished. However, it is prone to cause starvation for new peers because of their low initial reputation [Chen et al. 2010]. Compared to public/private P2P, agents in Thunder Crystal can stay online for long periods of time to earn money rewards without discriminating against new users.

Second, in Thunder Crystal, content caching on agents is more flexible by decoupling the cached content and the consumed content. In P2P networks, it is common for peers to cache content to be consumed by themselves, but it is difficult to motivate them to cache unrelated content, although some P2P VoD systems force users to cache related content with impaired user QoE [Wu et al. 2009]. In contrast, it is natural to decouple content caching and content consuming in Thunder Crystal because agents have the incentive to cache any content that can generate crystals. The cached content cannot be directly accessed by agents for consumption. The advantage of this is to apply more flexible active content pushing and content replication strategies to more efficiently serve user requests.

Last, it is expected that Thunder Crystal can provide better content distribution service with enhanced availability and more flexible content caching than P2P networks with increased management cost. In Thunder Crystal, the global information of the chunks replicated by each device is the prior knowledge for content replication and request scheduling, implying that the content management cost is significant if the system scales to millions of agent devices. Distributed content management at a low cost must be designed to ensure the scalability of Thunder Crystal, which will be further investigated in our future work. In fact, by involving more agents, Thunder Crystal can make a profit as long as the management cost function is linear or sublinear to the agent population.

4. MEASUREMENT EVALUATION OF THUNDER CRYSTAL

We present a measurement study of Thunder Crystal in this section, serving two purposes. First, Thunder Crystal heavily relies on agent devices for content distribution. How the agents behave (e.g., how much resource agents would like to contribute) determines Thunder Crystal's service capacity and service quality. Thus, it is necessary to study the stability of agent devices, the distribution of contributed resources, and other agent behavior. Second, it is unknown how the strategies proposed in the previous section perform in a real distributed and complicated system. Measurement studies can reflect how well they work in practice and help us discover system weaknesses for further improvement in future work.

4.1. Data Description

The data collected for this study are from December 1, 2014 through December 31, 2014. During this period, the number of active agent devices was about 11,000. Although it was not difficult to scale up the system by involving more agent devices, the concern was that the number of crystals earned by agents depends on the content distribution load.[10] If the agent is oversupplied, then the money paid to each agent may be insufficient to cover the costs or be too little to attract agents. Thus, based on the load of the current system, the number of agent devices was restricted to fewer than 13,000. Excluding some inactive agent devices, the number of daily active agent devices was about 11,000.

For data collection, a data report module was embedded into the software installed on agent devices, which reported necessary information to a log server farm. Reported information included the number of agent devices working at any moment, the measured instant uploading speed, event-driven messages recording the access log of content, and other activities. Among all reported data, fat agent devices accounted for about 25% and thin agent devices about 75%. These agent devices were located over all geographical regions and covered almost all ISPs in China. All together, we collected 35 billion reports with a size of approximately 3TB. The Hadoop platform was used to process this dataset for our measurement study and analysis.

4.2. Demand Pattern

Thunder Crystal is mainly used to support video downloading. Similar to other online systems, the user demand exhibits a strong weekly pattern, as shown in Figure 2. The x-axis shows the time from Monday to Sunday, and the y-axis shows the number of user requests. For each day, the trough point is achieved in the morning from 6:00am to 7:00am, whereas the peak point is achieved in the evening from around 8:00pm to 9:00pm. During weekends, there is an evident increase in user requests. Both the trough point and the peak point are lifted up. As revealed by the collected data, user requests increase by 10% to 20% during weekends. The system performance and the number of crystals earned by agents are heavily affected by the number of user requests, which will be discussed in detail later.

4.3. Resource Distributions

Serving user requests is equivalent to matching user demand with the resources contributed by agents. In Thunder Crystal, each agent needs to contribute two kinds of resources: upload bandwidth and storage. We measure the upload capacity and storage

[10]Thunder Crystal provides a commercial content distribution service for content providers. The content distribution load depends on the business volume.

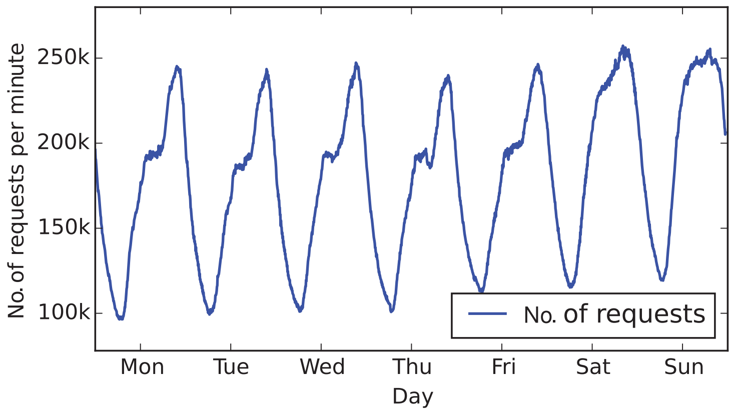

Fig. 2. Daily pattern of user requests for file downloading.

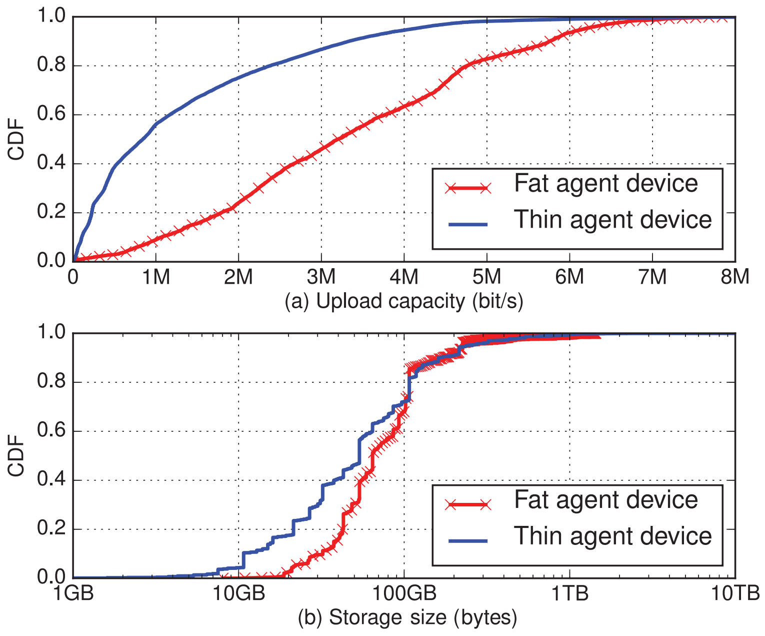

Fig. 3. CDF of upload capacity and storage size.

size of each device, and present the cumulative distribution function (CDF) of each kind of resource in Figure 3.

The upload capacity reported by agents is not reliable, as agents may set a very large value for the limit of upload rate. Therefore, the capacity is mainly determined by the Internet access plans purchased by agents. In Figure 3, the upload capacity is obtained by measuring each agent device's upload speed. Our approach is an approximate method to estimate each device's true upload capacity by recording each device's peak upload speed on a daily basis. For each device, 30 peak upload speeds will be recorded over a 1 month period, assuming that the device will be online every day. The median peak speed is taken as the device's upload capacity and plotted in Figure 3(a), in which we observe that around 60% of the thin agent devices have less than 1Mb/s upload capacity and that around 80% of the fat agent devices have no more than 5Mb/s upload capacity. In general, devices of fat agents are equipped with much more powerful network access capacity.

The storage size contributed by agents for content caching is easily measured and collected. The CDF curve of agent devices' storage size is plotted in Figure 3(b). We find that most agent devices allocate from 10GB to 100GB storage. Overall, fat agents contribute more storage for each device than thin agents. However, a considerable number of fat agents use less storage than thin agents, and the devices with extremely large storage (more than 20TB) are from thin agents with upload capacity less than 2Mb/s. It seems that thin agents with a slower uploading rate are more aggressive in boosting upload by caching more content. In fact, the amount of the cached content is decided by storage size together with download capacity. More discussion on this problem will be presented later.

The amount of storage that should be allocated for given bandwidth resources is unknown. We believe that agents use arbitrary strategies to decide the storage size. The fat agents with more powerful network capacity are expected to contribute larger storage size to cache more content to more efficiently utilize their upload capacity. We calculated the correlation between the upload capacity and the contributed storage size for thin agents and fat agents separately. Surprisingly, the correlation is 0.4025 for thin agents but only 0.064 for fat agents, suggesting that fat agents spend less effort on optimizing the storage resource for content caching.

4.4. Agent Behavior

The (bandwidth and storage) resource contributed by agents is not the only factor used to determine Thunder Crystal's service quality. The agent behavior, mainly referring to the agent devices' online and offline time, is also essential for a content distribution service. The longer an agent device is online, the more stable service the device provides. Thus, by examining how much time an agent device devotes to Thunder Crystal, we learn how well the agents can support content distribution.

Figure 4(a) shows the varying number of active devices during a single day. The number in the figure is between 11K and 11.4K, which is a rather small range. Figure 4(b), the plot of the CDF curve of devices' online duration time during 1 day, shows that most agents stay online the whole day and that fat agents' devices have longer online time. Figure 4(c) shows the active number of days of each agent. An agent is active as long as one of his devices is online. As we can see, more than 50% of the agents are active every day during the measured 1 month period. Based on the preceding observations, we conclude that Thunder Crystal can provide a stable content distribution service since most agents have strong motivation to stay online. Agent devices act as mini CDN servers instead of peers in P2P networks. Although the stability of a single agent device is not as good as a server, there are plenty of agent devices, which can work together. For popular content, a sufficient number of copies can be deployed on different devices so that Thunder Crystal can provide a content distribution service as well as that of CDNs. The trouble comes from those with unpopular content with few user requests. The efficiency of bandwidth utilization will be lowered dramatically if an agent device is taken up by unpopular content, which motivates us to study content popularity distribution and evolution in the next section.

4.5. Measurement of Content Popularity

The video popularity distribution significantly affects the content replication strategy [Zhou et al. 2015]. A better understanding of the popularity distribution will benefit the design of the content replication strategy. In this study, the percentage of user requests for a video on a particular day is regarded as its popularity on that day. This

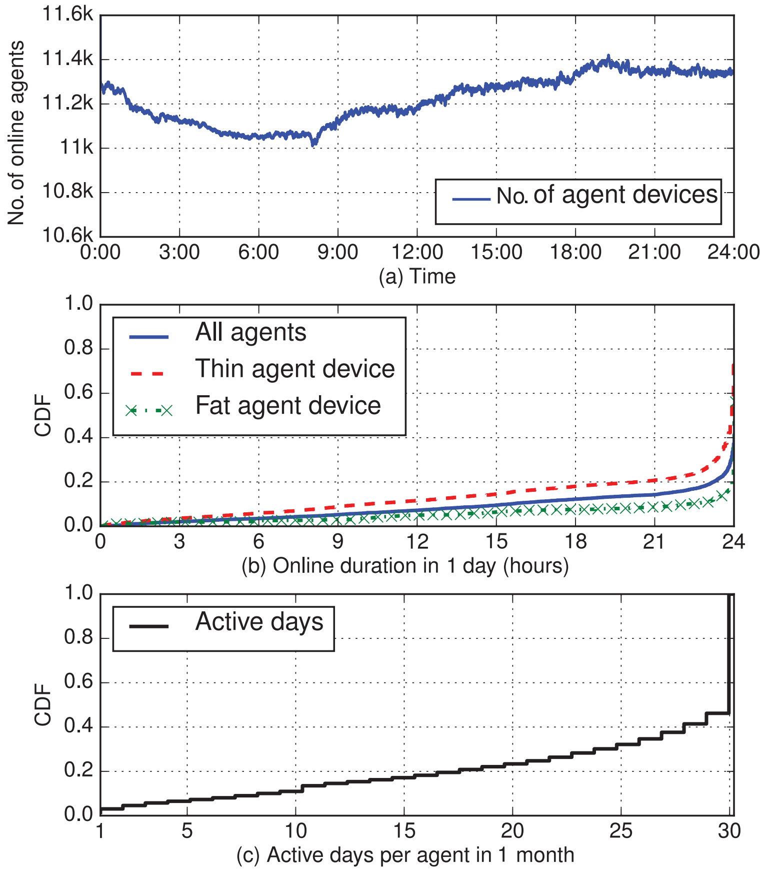

Fig. 4. Agent behavior observation.

study focuses on two facets: popularity skewness and popularity decay rate. Although there are several related works that have investigated the two facets in Internet-based content distribution systems, such as YouTube [Cha et al. 2007], the TV-on-Demand system [Abrahamsson and Nordmark 2012], and Tencent Video [Zhou et al. 2015], we still need to get the popularity characteristics of files distributed via Thunder Crystal to facilitate the design of the replication strategy.

Based on the collected information about the number of requests for each file in each day, we plot the popularity distribution of all files on a particular day, as shown in Figure 5(a). The file popularity follows the power law distribution with a long and heavy tail, which is consistent with previous works (e.g. Chen et al. [2014] and Zhou et al. [2015]). With heavy tail popularity distribution, it is difficult to provide distribution service for all videos by purely relying on agent devices, because even if each agent device can contribute 10 to 100GB storage size, it is still very limited compared to the size of all videos. To keep agent devices working efficiently, it is necessary to cache the most popular content instead of all videos on agent devices. Thus, the replication algorithm pushes out videos in descending order of popularity until the server bandwidth is exhausted. The degree of skewness in the video popularity distribution is very important for content pushing. If the distribution is more skewed with a lighter tail, most user requests will concentrate on fewer videos, implying that user requests can

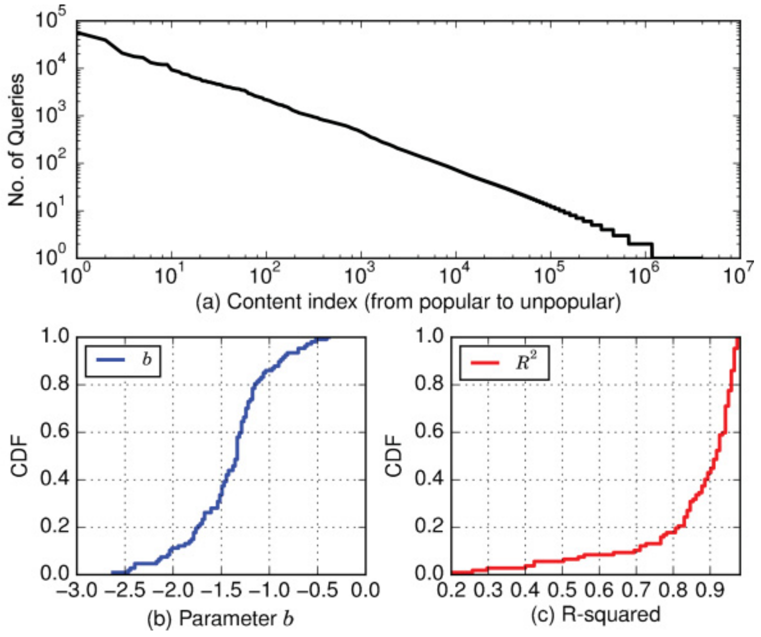

Fig. 5. Dynamics of content popularity.

be well served by caching a smaller number of videos. Otherwise, more videos must be covered to serve the same fraction of user requests, which may lower the efficiency of agent devices.

In addition to the skewness of popularity distribution, video popularity evolution (i.e., the fact that users lose interest in video content very quickly) also affects the pushing strategy significantly. To study how fast video popularity declines, we use the equation $f(x) = ax^b$ to fit the trace of the request number for each file over 30 days,[11] once the file is pushed into Thunder Crystal. Here, parameter b tells us how fast file popularity declines. The CDF curves of all files' b and R-squared results are plotted in Figure 5(b) and (c). As we can see, b is less than -1 for most files, implying fast popularity decline. The fitting results indicate that agent devices need popular content for uploading, and the out-of-date content must be replaced by new content in time. Therefore, a set of Thunder servers is set up to push the latest video content per day to agent devices to keep agents alive. Note that the value b is different with different videos, implying that the popularity decline is divergent and the popularity of each video must be recalculated on a daily basis for content replication.

4.6. Measurement of Content Pushing

Recall that the content pushing strategy randomly selects a free agent device to which it will push content for each unit time. Files are pushed out to agents if their deployed numbers of copies are less than the target numbers in the descending order of popularity. For a particular agent device, the content updating rate depends on the device's downloading rate. With a faster downloading rate, the odds that the agent device is selected by the Thunder servers are higher, as only free agent devices without downloading tasks will be considered for content pushing. Figure 6 shows the number of file copies pushed to thin or fat agents on each day. The black/white column represents

[11]The equation $f(x) = ax^b$ achieves smaller fitting errors than exponential equation or quadratic equation.

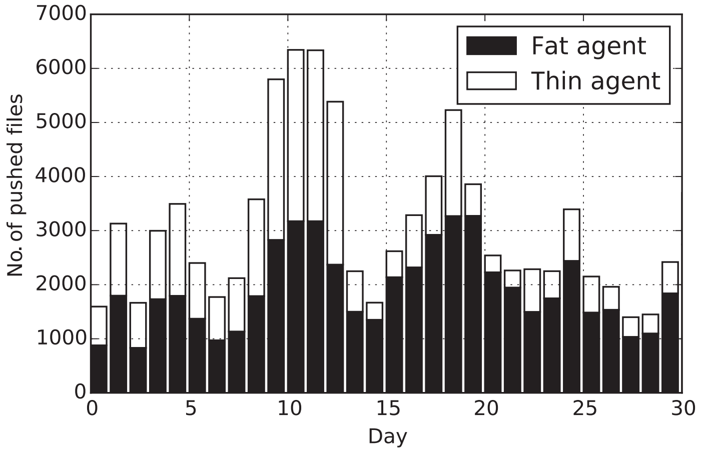

Fig. 6. Number of copies pushed to fat or thin agents on each day.

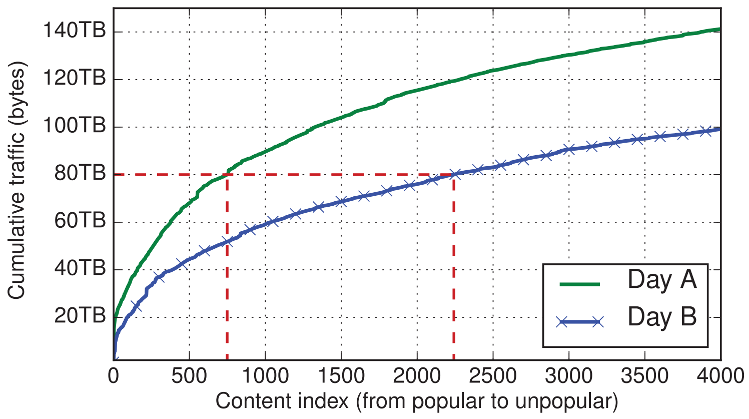

Fig. 7. Cumulative traffic in the content deployment.

the portion of files taken by fat/thin agents. As we can observe, although only 25% of devices belong to fat agents, more than 50% of files copies are taken by them due to their faster downloading rates. Note that the total upload traffic of Thunder Crystal is fixed at 80TB, but the number of total pushed copies oscillate wildly, as the file size could vary in a wide range.

Due to limited resources (i.e., limited upload capacity of Thunder servers and limited storage space on agent devices), it is not a wise choice to push all content, both popular and unpopular, to agents. Instead of covering all content through agent devices, Thunder Crystal pushes files in the descending order of popularity. According to the content pushing strategy, a less popular file will not be pushed by Thunder servers unless all files with higher popularity have been deployed with sufficient copies. Intuitively speaking, Thunder Crystal deploys more popular videos with higher priority. The files included in the set are affected by the skewness of popularity distribution significantly. If the popularity distribution is more skewed (i.e., more users requesting fewer files), then fewer files but more replicas for each file are required. On the contrary, if the popularity is less skewed, more videos but fewer replicas of each video file are requested. Figure 7 illustrates this problem by comparing the content pushing

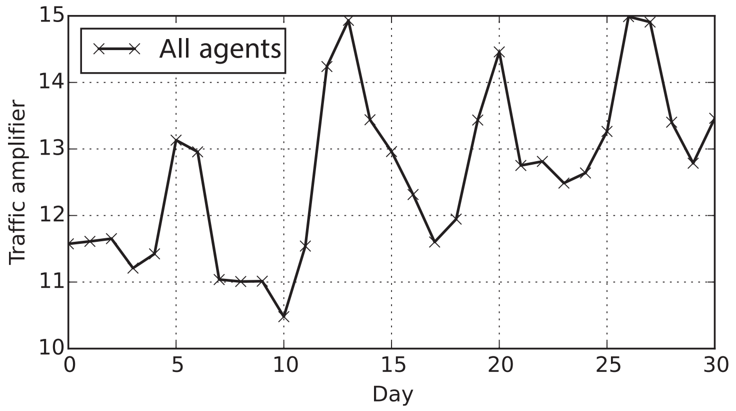

Fig. 8. System traffic amplifier in 1 month.

results of Day A to more skewed popularity distribution and Day B with less skewed popularity distribution. The top 749 files on Day A and the top 2,242 files on Day B are pushed by Thunder servers.

4.7. System Efficiency

How to evaluate the performance of Thunder Crystal system is the most basic question that needs to be answered by our work. In principle, the Thunder Crystal system scales its capacity to distribution content through a large population of agent devices. The number of times the capacity can be amplified by agent devices is of great importance in Thunder Crystal. The Thunder servers push a total of 80TB of traffic to agent devices per day. The total daily traffic contributed by all agents reflects how efficiently the bandwidth is utilized. Therefore, we define amplification of upload traffic at Day t as $\gamma_t = \frac{\sum T_i(t)}{80TB}$, where $T_i(t)$ is the device i's upload traffic on Day t. For file downloading, we believe that the traffic amplifier is a telling metric, indicating the downloading load that can be taken by Thunder Crystal. The values of γ_t over 30 days are plotted in Figure 8.

As we can observe, the traffic is scaled about 10 to 15 times in most cases, indicating the effectiveness of the crowdsourcing-based content distribution. Interestingly, the traffic amplifier reaches a peak value about every 5 days. Actually, these peak values correspond to weekend days. As we stated regarding the demand pattern in Figure 2, the user requests on weekends will be inflated by 10% to 20%, resulting in the increase in the traffic amplifier.

In addition to the traffic amplifier, we propose the second metric, system efficiency, to measure how efficiently the bandwidth resource of agent devices is utilized. The system efficiency is defined as $\eta = \frac{C_s}{\sum_i C_i}$. Here, C_i is the upload capacity of device i, implying that $\sum_i C_i$ is the upload capacity of the whole system, and C_s is the measured upload capacity of the whole system. $\eta \times \sum_i C_i$ indicates the maximum instant uploading speed of the whole system. From the definition of η, we believe that η is a better metric to evaluate the system capacity for video streaming with inelastic bandwidth demands. The value of η is measured on a daily basis and plotted in Figure 9 over 30 days.

From the results in Figure 9, we can conclude that during peak hours, the system efficiency is 0.6 to 0.7, implying that in theory the maximum streaming rate of the whole system is about 0.6 to 0.7 of the whole system's capacity. Compared to the traffic amplifier presented in Figure 8, the variation range of system efficiency is quite small, which can be interpreted as follows. The system efficiency depends on the

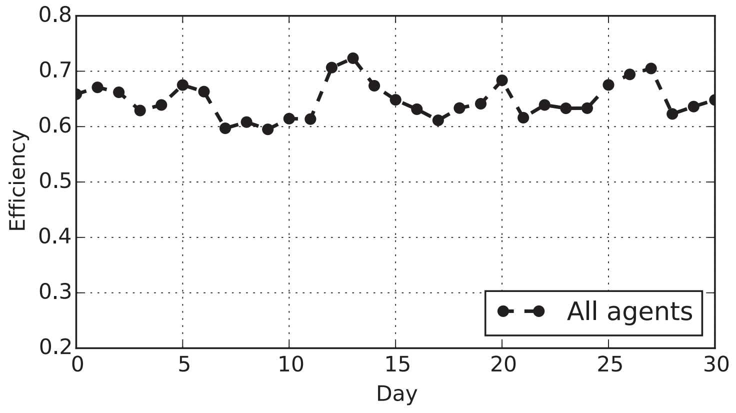

Fig. 9. System efficiency in 1 month.

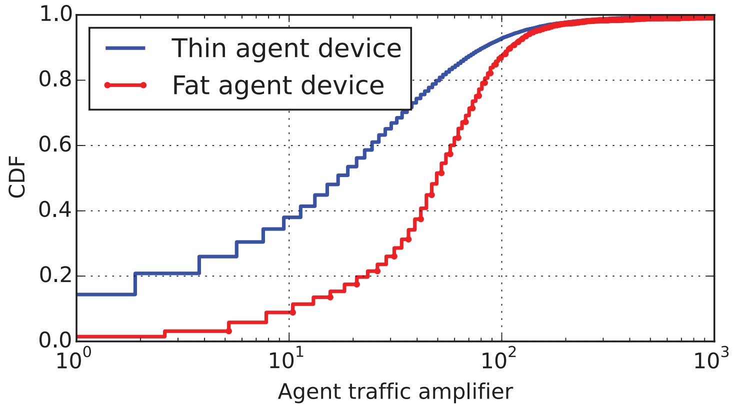

Fig. 10. CDF curves of the traffic amplifier for thin and fat agents.

maximum instant uploading speed during peak hours, which is mainly determined by available bandwidth that can be crawled from each agent. As long as the upload capacity of an agent device does not oscillate sharply, the system efficiency will not change dramatically.

4.8. Agent Efficiency

Since the atomic unit is the agent device, in this section we investigate the efficiency of each agent. Similar to the metrics used to evaluate system performance, we define the traffic amplifier for each agent device. For a particular agent device i, the traffic amplifier is $\gamma_t^i = \frac{T_i(t)}{D_i(t)}$, where $T_i(t)$ is the total upload traffic and $D_i(t)$ is the total download traffic on a particular day t for device i.

The CDF curves of the traffic amplifier for thin agents and fat agents are plotted separately in Figure 10. However, as shown in this figure, fat agents achieve much higher traffic amplifier than thin agents. Upload traffic amplifier is a ratio of upload/download traffic, where the download traffic is from the content pushed by Thunder servers. Although fat agents have a more powerful upload capacity, we hope that fat and thin agents should achieve a comparable traffic amplifier to efficiently utilize the content pushed by Thunder servers. Thus, we are interested in exploring the reason for the result in a lower traffic amplifier for thin agents.

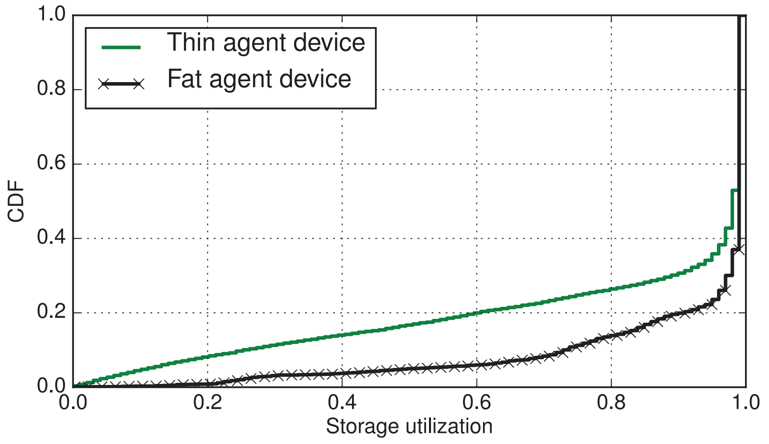

Fig. 11. Storage utilization for thin and fat agents.

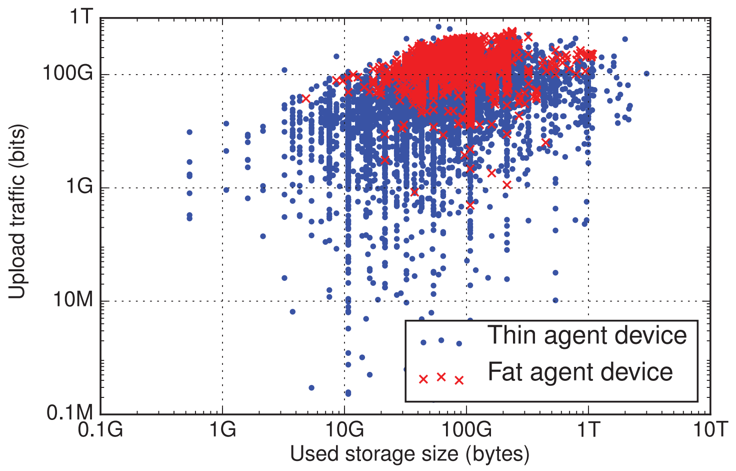

Fig. 12. Scatter plot of used storage size and upload traffic.

Since the content is cached on agent devices before it is uploaded, we speculate that storage is the key factor in determining the traffic amplifier. Recall that in Figure 3, thin agents contribute storage resources very aggressively. However, it is not reasonable to simply use the storage size to evaluate the upload traffic. In fact, the upload traffic of a particular agent device is mainly determined by the used storage, which in turn is mainly decided by the download capacity. Figure 11 shows the CDF curve of the storage occupied by content pushed from Thunder servers for thin and fat agents. Although fat agents contribute more storage as shown in Figure 3, bigger portions of storage are occupied by pushed content, because the probability of selecting fat agents by Thunder servers is higher because of their higher downloading rate.

To further validate the relationship between upload traffic and occupied storage, we plot the values of all devices, each as a point, in Figure 12 with the x-axis representing used storage and the y-axis representing upload traffic. As expected, there is a positive correlation (about 0.4) between upload traffic and used storage.

Based on the preceding discussions, we conclude that fat agents have the potential to further improve the traffic amplifier by allocating more storage for the content cache. There are two reasons to support our opinion. First, fat agents do not contribute storage proportional to upload capacity, as shown in Figure 3. The correlation between upload capacity and storage size is 0.064, which is much lower than the 0.4025 for thin agents. Second, Figure 11 shows that the storage resource of fat agents is more exhaustive than that of thin agents. According to our measurement study, equipped with powerful download/upload capacity, fat agents can cache more content if more storage is available, which will result in a higher traffic amplifier.

5. DISCUSSION

Overall, the measurement study shows that Thunder Crystal works well in practice; however, in the first version, we tentatively implemented many heuristic strategies. However, to extend the system scale and understand the performance gap and how much we can improve, it is necessary to study the Thunder Crystal system theoretically. In this section, we raise three open problems that we believe are essential to improving system performance. Solving these open problems could be rather difficult, as Thunder Crystal is a large-scale complicated system. These problems will be studied further in future work. In this article, we briefly discuss them and point out their significance for Thunder Crystal:

—The content pushing strategy determines the number of pushed copies for each file. If the number of copies for any file is insufficient or the number of copies for a file is oversupplied, the agent devices may starve to earn crystals, due to the lack of user requests. Thunder Crystal makes the push decision for file j by the empirical equation $M_j = (\beta N_j)^\alpha$, where $\alpha = 0.96$ and $\beta = 0.05$. N_j is the number of requests for file j (i.e., the historical popularity of file j). α is determined by how fast the file popularity declines, whereas β decides the range of files covered by Thunder Crystal. With a fast popularity decay rate, a lower α should be set. With a larger β, fewer files will be covered by Thunder Crystal. By fitting with historical data, we decide to set the α and β as specified previously. The optimal content pushing strategy should synthetically consider three factors: the rate at which file popularity declines, upload capacity of agents replicating the file, and the popularity of the files. A theoretical model or analysis is expected to quantify the gap between any strategy and the optimal strategy, even if solving the optimal strategy could be NP-complete.

—Agents are motivated by monetary rewards to contribute bandwidth resources. The bandwidth pricing should be reasonable, as otherwise the earnings of either Thunder Crystal or the agents will be reduced. In the current version, the bandwidth pricing is time dependent to encourage agents to contribute more bandwidth resources during peak hours. More specifically, from 12:00 to 22:00, 1GB upload traffic is equivalent to one crystal, but only half a crystal otherwise. A more sophisticated bandwidth pricing strategy is called for so that the monetary rewards for agents are not only based on upload traffic but also based on agents' upload capacity, stability, storage, and so on, to achieve a better incentive effect.

—A fairness balance strategy is needed to ensure the fairness of the system. In Thunder Crystal, agents can decide on the online/offline time, contributed storage size, and limits of the uploading rate. Each agent may use a strategy to make these decisions, with the aim to maximize their own net profits. In a fair system, the more resources an agent contributes, the more crystals he earns. There are many random events that could impair fairness and discourage agents in contributing resources, such as when the agent devices replicating an extremely popular file may earn considerably more crystals than the others without replicating the file, or when the request scheduling strategy randomly redirects user requests to agent devices; lucky agents may be assigned many more user requests and earn more crystals. Thus, similar to the load balancing strategy for balancing the load among edge servers considered in

CDNs [Pallis and Vakali 2006], a fairness balance strategy is needed in Thunder Crystal.

6. RELATED WORK

Crowdsourcing is a novel angle for system design, quality evaluation, system efficiency improvement, and so on. It is very common that systems providing Internet services are large scaled and distributed. Crowdsourcing-based solutions are more efficient and effective in accomplishing challenging distributed tasks. Many studies on crowdsourcing have been done in recent years. Our previous work [Chen et al. 2015] was the first to introduce how to build a real crowdsourcing-based content distribution platform. MicroCast was designed and implemented by utilizing the resources on smartphones within the same group in a cooperative way to improve the video streaming experience [Keller et al. 2012]. A measurement study was conducted to validate the advance of MicroCast. A crowdsourceable framework was proposed to quantify the QoE of multimedia content [Wu et al. 2013] by asking Internet users to conduct QoE experiments on their own computers. Similarly, a crowdsourcing-based approach was proposed for quantification of YouTube QoE [Hoßfeld et al. 2011]. A crowdsourcing strategy was proposed to characterize ISP services by utilizing network-intensive applications running on end systems [Bischof et al. 2011], of which the feasibility was proved by collecting the traffic data from BitTorrent users. The crowdsourcing-based system Portolan aims to monitor and measure large-scale networks using smartphones as mobile observation elements. Portolan is able to obtain the graph of the Internet and associate performance indexes (received signal strength, maximum throughput) of cellular networks to geographic locations [Faggiani et al. 2013]. The multimedia service (e.g., the live video streaming service) is shifting from a conventional single source to a crowdsource. Twitch is one of such crowdsourced live streaming systems. Its architecture is outlined by using the crawled data and captured traffic of local broadcasters/viewers [Zhang and Liu 2015]. However, except for our work, there is no existing work addressing the design and implementation of a crowdsourcing-based content distribution platform.

CDN and P2P are the two most prevalent architectures for accelerating content distribution, including video streaming and video downloading. CDN systems deploy a large number of dedicated edge servers everywhere to ensure QoS [Huang et al. 2008b], resulting in a higher cost than that of P2P networks. System performance can be improved by developing more advanced user request scheduling and content caching strategies [Krishnan et al. 2009; Traverso et al. 2013]. Compared to pure CDN systems, P2P systems are much more cost efficient, which have been studied from theoretical analysis [Zhou et al. 2013; Tan and Massoulié 2013] to practical system design and evaluations [Liu et al. 2010b; Huang et al. 2008a]. A hybrid CDN/P2P approach [Yin et al. 2009] is the solution proposed to achieve the scalability and low cost advantages of P2P along with the reliability and manageability of CDNs. Cost reduction is a major concern in such systems [Balachandran et al. 2013]. However, simply combining P2P with CDN cannot effectively address all P2P weaknesses (e.g., unstable online time, scarce storage for replication, and criticism for possible infringement of content copyright). Different from CDN or P2P, Thunder Crystal is a novel solution to build the infrastructure by renting resources from agents. Although it shares some similarity with CDN and P2P (e.g., the content replication and pushing strategies are also considered in CDN and P2P), it achieves advantages of lower cost than CDN and more reliable bandwidth supply than P2P, showing its prospects.

How to motivate peers to contribute bandwidth resources to assist in content distribution has been studied extensively in P2P networks. Various peer incentive mechanisms have been designed toward realizing a perfect private BT file sharing system. The ideas surrounding these mechanisms can be borrowed to design the incentive mechanism in Thunder Crystal. For example, a novel incentive mechanism was proposed by rewarding each peer based on its dedicated upload bandwidth [Wu et al. 2014]. Zhao et al. [2012] created a general framework to analyze various incentive protocols and the system state evolution with multiple incentive protocols executed by peers simultaneously. Yang and Lou [2012] proposed a profitable business model to analyze the benefits for content service providers, ISPs, and end users in a competing market. Rahman et al. [2011] outlined the design space of incentive mechanisms in distributed file sharing systems and conducted a simulation-based analysis to analyze the strengths and weaknesses of each incentive mechanism. Jia et al. [2013] showed that both credit-based and sharing ratio enforcement policies in private BT systems can lead to system-wide "crunches" or "crashes," where the system seizes completely due to too little or too much credit, respectively. All of the systems mentioned previously are built based on virtual currency. No matter how much effort a peer contributes, only virtual money is earned, which may inhibit the contribution of powerful peers and inhibit the demands of weak peers due to the lack of virtual credits. However, Thunder Crystal is built by providing cash rewards as a more effective incentive mechanism to motivate agents.

7. CONCLUSION

In this article, we introduced the design principles, implementation details, and measurement study of Thunder Crystal, a crowdsourcing-based content distribution platform. Thunder Crystal rents resources from agents to mainly support video downloading. Monetary rewards are paid to agents based on agents' upload traffic. Three key strategies—content replication, content pushing, and request scheduling and chunk transmission—are proposed to operate the system. A measurement study was conducted to validate the effectiveness of Thunder Crystal. We believe that crowdsourcing is a promising method for building a content distribution infrastructure, which is evident from fact that the content distribution cost can be reduced greatly by Thunder Crystal. With a novel crowdsourcing-based incentive mechanism, our work shows the feasibility and effectiveness of Thunder Crystal. Finally, significant future efforts are required to enhance the efficiency of Thunder Crystal, as stated in our discussion of open problems.

REFERENCES

Henrik Abrahamsson and Mattias Nordmark. 2012. Program popularity and viewer behaviour in a large TV-on-Demand system. In *Proceedings of the 2012 ACM Conference on Internet Measurement*. ACM, New York, NY, 199–210.

Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. 2012. Unreeling Netflix: Understanding and improving multi-CDN movie delivery. In *Proceedings of the 2012 Annual IEEE International Conference on Computer Communications*. IEEE, New York, NY, 1620–1628.

Athula Balachandran, Vyas Sekar, Aditya Akella, and Srinivasan Seshan. 2013. Analyzing the potential benefits of CDN augmentation strategies for Internet video workloads. In *Proceedings of the 2013 ACM Conference on Internet Measurement*. ACM, New York, NY, 43–56.

Zachary S. Bischof, John S. Otto, Mario A. Sánchez, John P. Rula, David R. Choffnes, and Fabián E. Bustamante. 2011. Crowdsourcing ISP characterization to the network edge. In *Proceedings of the 1st ACM SIGCOMM Workshop on Measurements Up the Stack*. ACM, New York, NY, 61–66.

Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. 2007. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 2007 ACM Conference on Internet Measurement*. ACM, New York, NY, 1–14.

Liang Chen, Yipeng Zhou, and Dah Ming Chiu. 2014. A lifetime model of online video popularity. In *Proceedings of the 2014 Conference on Computer Communication and Networks*. IEEE, Los Alamitos, CA, 1–8.

Liang Chen, Yipeng Zhou, Mi Jing, and Richard T. B. Ma. 2015. Thunder Crystal: A novel crowdsourcing-based content distribution platform. In *Proceedings of the 2015 ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, New York, NY, 43–48.

Xiaowei Chen, Yixin Jiang, and Xiaowen Chu. 2010. Measurements, analysis and modeling of private trackers. In *Proceedings of the 2010 IEEE 10th International Conference on Peer-to-Peer Computing*. IEEE, Los Alamitos, CA, 1–10.

Cisco. 2012. *Cisco Visual Networking Index: Forecast and Methodology, 2011–2016*. White Paper. Cisco, San Jose, CA.

Bram Cohen. 2003. Incentives build robustness in BitTorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Vol. 6. 68–72.

Adriano Faggiani, Enrico Gregori, Luciano Lenzini, Valerio Luconi, and Alessio Vecchio. 2013. Network sensing through smartphone-based crowdsourcing. In *Proceedings of the 2013 ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, 1–2.

Bin Fan, Dah-Ming Chiu, and John Lui. 2006. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *Proceedings of the 2006 IEEE International Conference on Network Protocols*. IEEE, Los Alamitos, CA, 239–248.

Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. 2011. Quantification of YouTube QoE via crowdsourcing. In *Proceedings of the 2011 IEEE International Symposium on Multimedia*. IEEE, Los Alamitos, CA, 494–499.

Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. 2008b. Measuring and evaluating large-scale CDNs. In *Proceedings of the 2008 ACM Conference on Internet Measurement*, Vol. 8. 15–29.

Yan Huang, Tom Z. J. Fu, Dah-Ming Chiu, John Lui, and Cheng Huang. 2008a. Challenges, design and analysis of a large-scale P2P-VoD system. *ACM SIGCOMM Computer Communication Review* 38, 4, 375–388.

Adele Lu Jia, Raziur Rahman, Tamas Vinko, Johan A. Pouwelse, and Dick H. J. Epema. 2013. Systemic risk and user-level performance in private P2P communities. *IEEE Transactions on Parallel and Distributed Systems* 24, 12, 2503–2512.

Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. 2012. MicroCast: Cooperative video streaming on smartphones. In *Proceedings of the 2012 International Conference on Mobile Systems, Applications, and Services*. ACM, New York, NY, 57–70.

Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. 2009. Moving beyond end-to-end path information to optimize CDN performance. In *Proceedings of the 2009 ACM Conference on Internet Measurement*. ACM, New York, NY, 190–201.

Zhengye Liu, Prithula Dhungel, Di Wu, Chao Zhang, and Keith W. Ross. 2010a. Understanding and improving ratio incentives in private communities. In *Proceedings of the 2010 IEEE International Conference on Distributed Computing Systems*. IEEE, Los Alamitos, CA, 610–621.

Zimu Liu, Chuan Wu, Baochun Li, and Shuqiao Zhao. 2010b. UUSee: Large-scale operational on-demand streaming with random network coding. In *Proceedings of the 2010 Annual IEEE International Conference on Computer Communications*. IEEE, Los Alamitos, CA, 1–9.

George Pallis and Athena Vakali. 2006. Insight and perspectives for content delivery networks. *Communications of the ACM* 49, 1, 101–106.

Andrea Passarella. 2012. A survey on content-centric technologies for the current Internet: CDN and P2P solutions. *Computer Communications* 35, 1, 1–32.

Rameez Rahman, Tamás Vinkó, David Hales, Johan Pouwelse, and Henk Sips. 2011. Design space analysis for modeling incentives in distributed systems. *ACM SIGCOMM Computer Communication Review* 41, 4, 182–193.

Bo Tan and Laurent Massoulié. 2013. Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM Transactions on Networking* 21, 2, 566–579.

Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. 2013. Temporal locality in today's content caching: Why it matters and how to model it. *ACM SIGCOMM Computer Communication Review* 43, 5, 5–12.

Chen-Chi Wu, Kuan-Ta Chen, Yu-Chun Chang, and Chin-Laung Lei. 2013. Crowdsourcing multimedia QoE evaluation: A trusted framework. *IEEE Transactions on Multimedia* 15, 5, 1121–1137.

Di Wu, Chao Liang, Yong Liu, and Keith Ross. 2009. View-upload decoupling: A redesign of multi-channel P2P video systems. In *Proceedings of the 2009 International Conference on Computer Communications*. IEEE, Los Alamitos, CA, 2726–2730.

Weijie Wu, Richard T. B. Ma, and John C. S. Lui. 2014. Distributed caching via rewarding: An incentive scheme design in P2P-VoD systems. *IEEE Transactions on Parallel and Distributed Systems* 25, 3, 612–621.

Libin Yang and Wei Lou. 2012. Pricing, competition and innovation: A profitable business model to resolve the tussle involved in peer-to-peer streaming applications. In *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*. 1–9.

Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. 2009. Design and deployment of a hybrid CDN-P2P system for live video streaming: Experiences with LiveSky. In *Proceedings of the 2009 International Conference on Multimedia*. ACM, New York, NY, 25–34.

Cong Zhang and Jiangchuan Liu. 2015. On crowdsourced interactive live streaming: A Twitch.TV-based measurement study. In *Proceedings of the 2015 ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, New York, NY, 55–60.

Bridge Qiao Zhao, John C. S. Lui, and Dah-Ming Chiu. 2012. A mathematical framework for analyzing adaptive incentive protocols in P2P networks. *IEEE/ACM Transactions on Networking* 20, 2, 367–380.

Yipeng Zhou, Liang Chen, Chunfeng Yang, and Dah Ming Chiu. 2015. Video popularity dynamics and its implication for replication. *IEEE Transactions on Multimedia* 17, 8, 1273–1285.

Yipeng Zhou, Tom Z. J. Fu, and Dah Ming Chiu. 2013. On replication algorithm in P2P VoD. *IEEE/ACM Transactions on Networking* 21, 1, 233–243.

Yipeng Zhou, Tom Z. J. Fu, and Dah Ming Chiu. 2015. A unifying model and analysis of P2P VoD replication and scheduling. *IEEE/ACM Transactions on Networking* 23, 4, 1163–1175.

Received August 2015; revised May 2016; accepted July 2016