Thunder Crystal: A Novel Crowdsourcing-based Content Distribution Platform

Liang Chen[†], Yipeng Zhou[*], Mi Jing[‡], Richard T.B. Ma[^]
[†]College of Information Engineering, Shenzhen University
[*]College of Computer Science and Software Engineering, Shenzhen University
[‡]Xunlei Network Technology Company
[^]School of Computing, National University of Singapore
lchen@szu.edu.cn, ypzhou@szu.edu.cn, jingmi@xunlei.com, tbma@comp.nus.edu.sg

ABSTRACT

Content distribution, especially the distribution of video content, unavoidably consumes bandwidth resource heavily. Internet content providers (ICP) spend lots of money to buy content distribution network (CDN) service. By deploying thousands of edge servers close to end users, CDN companies are able to distribute content efficiently. In lieu of traditional CDN systems, we implement a crowdsourcing-based content distribution system, *Thunder Crystal*, which utilizes agents' upload bandwidth to amplify the content distribution capacity. Agents are well motivated to contribute storage and upload bandwidth to the system by rebated cash. As far as we know, this is a novel system that has not been studied before. In this work, we will present its design principles first. Then, we study agent behavior and methods to evaluate system efficiency and user efficiency. We evaluate the system by simulations, and observe that agents are well motivated to keep online most of the time and amplify the content distribution capacity by 10∼20 times.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*network management, network monitoring*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Measurement, Experimentation, Reliability

Keywords

Crowdsourcing, Content Distribution Networking

[*]Yipeng Zhou is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first p age. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a f ee. Request permissions from Permissions@acm.org.
NOSSDAV '15, March 18-20, 2015, Portland, OR, USA
Copyright 2015 ACM 978-1-4503-3352-8/15/03 $15.00
http://dx.doi.org/10.1145/2736084.2736085 .

1. INTRODUCTION

Content distribution over Internet is one of the most popular services. However, the delivery of (ultra high definition) videos extremely consumes bandwidth resource. In most cases, ICPs resort to CDN or peer-to-peer (P2P) networks to distribute content [9]. It either brings technology challenges (by using P2P) or incurs high bandwidth expenses (by using CDN) for ICPs.

Both CDN and P2P have been extensively studied, which are only introduced briefly here. CDN service providers deploy thousands of edge servers that are close to end users to reduce transmission delay. Each edge server is equipped with certain amount of storage to cache content. Once there is any user requesting for some content, CDN will direct the user's request to the nearest available edge server to deliver the content. Cached content on edge servers are centrally managed and periodically updated. The service quality can be guaranteed by deploying plenty of edge servers. However, it is expensive to buy CDN services. Alternatively, P2P is an economic way for content distribution. P2P allows peers downloading the same file to build collaborations and exchange partially downloaded content with each other. Although it has many advantages on reducing expenses, P2P at least has some weaknesses as follows. Firstly, user experience could be harmed as users upload and download content simultaneously. Secondly, P2P is a large scale distributed system, which is difficult for management and optimization. Thirdly, for unpopular content, it is not easy for a user to find other users for exchange content.

To address the issues mentioned above, we propose a hybrid system, Thunder Crystal, that takes advantages of both CDN and P2P. In Thunder Crystal, Thunder servers keep pushing content to agents; while agents are encouraged to contribute their storage (to cache content) and upload bandwidth (to distribute content to end users) by rebated cash from Thunder Crystal. Each agent is like a mini-CDN edge server, which is very close to end users. Most agents are normal Internet users, who would like to offer their surplus bandwidth with very low charge. Compared to the charging policies of the CDN companies, the cash rebated to agents is much lower, implying lower expenses for ICPs. In a word, Thunder Crystal acts as a crowdsourcing system, which offloads content distribution tasks to thousands of agents and gains profit for serving ICPs. In this paper, we will present the design principles of Thunder Crystal. We propose metrics to evaluate Thunder Crystal and agents, and conduct measurement study. As far as we know, this is the first work about crowdsourcing-based content distribution platform.

In the following parts, the background about agent and agent devices is introduced in Sec. 2. The whole system architecture and the function components are discussed in Sec. 3. The measurement results are presented in Sec. 4. We discuss the related works in Sec. 5 and conclude this paper in Sec. 6.

2. BACKGROUND

In Thunder Crystal, agents play very important role, who cache content on their devices and upload cached content to the end users. They are the basic units to complete content distribution tasks. In most cases, agents are normal Internet users who would like to get some returns through contributing residual bandwidth and storage resource. Normally, each agent can use at most 5 devices (that could be set up in different locations) to join the crowdsourcing based content distribution service. For some powerful agents, each of them can use at least 30 devices. The former type is referred as *thin agent*, and the later type is referred as *fat agent* for the rest discussion. Agents get crystals by uploading content to end users[1]. The value of each crystal is enough to cover the power and storage cost so that agents have motivation to take part in the game.

Typical agent devices are smart (Internet) access points (AP) and personal computers (PC). Smart APs are a kind of new emerging devices providing Internet access service (as well as wireless AP function). They are equipped with embedded operating system and large storage, typically around 1TB [1], so that they can complete some tasks independently. Compared with PCs, smart AP has smaller size, much lower price and less power consumption. With these advantages, smart APs can very efficiently download and upload content all day long. Thus, smart APs are ideal agent devices joining the services of Thunder Crystal. In current system, smart APs take more than 40% of total agent devices, but we expect that more and more smart APs will be used in the future.

3. SYSTEM DESIGN

In this section, we show the whole picture of the Thunder Crystal system. We firstly introduce the system architecture before we discuss the detailed strategies in Thunder Crystal.

3.1 Major components of the system

In Thunder Crystal, there are mainly three components: Thunder servers, agents and ICPs. Their roles are described as below:

1) *Thunder servers* keep pushing the latest content to agents' devices since Internet content loses user interests very fast. If content is not updated timely, agents cannot serve users very well. Thunder servers' bandwidth cost is Thunder Crystal's major expense.

2) *Agents* cache content pushed from Thunder servers to serve end users. During busy hours, agents can earn one crystal by uploading 1GB content. Thunder Crystal will direct user requests to agents caching requested content; while agents serve user requests with best efforts. Most agents are normal Internet users, and their bandwidth resource is shared with daily usage of other Internet services. Thus, agents are allowed to set the storage size and bandwidth used for Thunder Crystal freely.

[1]0.07 RMB is rebated to users for per crystal via Internet payment.

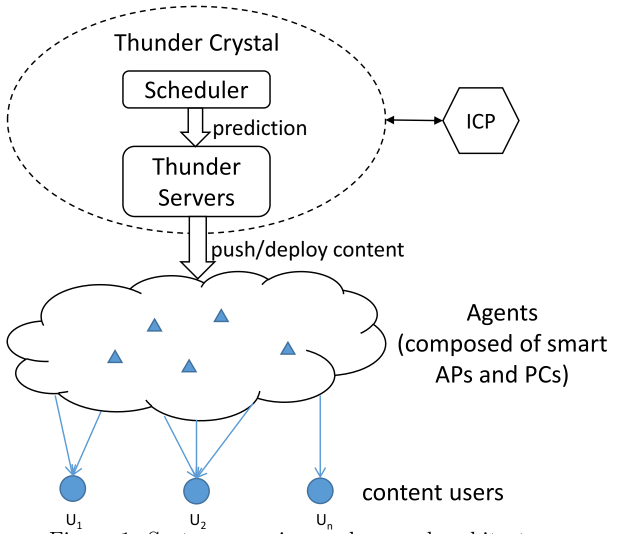

Figure 1: System overview and general architecture.

3) *ICPs* own the content and buy the service of Thunder Crystal to distribute content. Compared with traditional CDN companies, Thunder Crystal provides a lower price since much cheaper bandwidth are gathered from agents for content delivery.

The three components and how they interact are shown in Fig. 1. In this figure we can find that three strategies are needed to make the system work: 1) how to replicate content on agent devices; 2) how to push the latest content to different agents; 3) how to direct user requests to agent devices and how to transmit content from agents to end users. In Thunder Crystal, most content is high definition videos with size larger than 1GB. To solve the above problems, we need to cut and re-assemble files. We summarize the content units and their design purposes in Table 1. We will introduce the used content units and design principles in detail in the discussion of each strategy.

Table 1: 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

3.2 Content replication strategy

It is dangerous to replicate a complete unencrypted file on agents' devices. On one hand, agents could abuse cached content and infringe content's copyright. The content could be delivered to others by agents without notifying the content owners. On the other hand, agents could easily generate fake download if the replicated content is known. For example, an agent could be a home user who also downloads content for consuming. If the cached content is known, the agent could generate infinite number of requests to download the cached content to earn crystals.

To stop content abuse and fake download, each file is split into multiple *chunks* with 300MB for each chunk. Chunks are encrypted before they are pushed out to different agents. Each file is split into at least two chunks in case any agent own the complete file. For example, a 500MB file will be split into two chunks, one with 300MB and the other one with 200MB; while if a file is 220MB, it is still cut into two chunks with each 110MB. On agent devices, the chunks are reorganized and assembled into *segments* with 512MB for each segment. For agents, only the segments are seeable. In other words, there are two index mechanisms, segment index and chunk index. For agents, they can only find segments in their devices. For the Thunder Crystal, the user requests are directed to the agents with target chunks using chunk index.

3.3 Pushing strategy

To maintain the system efficiency, Thunder servers need to push out the latest content continuously to each device since the content system is open and highly dynamic. According to our measurement, the content popularity decays quite fast with time since content consumers' attention mainly focuses on the newly created content. Thus, the out-of-date content will lose user interests very soon. To make agents have available content to upload, Thunder servers should push the most popular content to agents continuously.

However, the bandwidth used for pushing is limited, we need a strategy to determine the number of pushed replicas for each file. For file j, if the number of requests to download file j is N_j in last day, then we will maintain $M_j = (0.05N_j)^\alpha$ replicas for file j. Here, $\alpha = 0.96$. 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. Thunder servers can push 80TB traffic per day as their traffic budget (since it is a cost for pushing content from servers to smart APs). Files will be pushed out by decreasing order of popularity until 80TB traffic is exhausted. In other words, more popular files will be pushed with higher priority than less popular files. If an agent's storage is full, the files with $m_j > M_j$ will be removed to receive new replicas.

In our pushing strategy, agent devices are not discriminated. Thunder servers push the new replicas randomly to different agent devices. Designing a more sophisticated pushing strategy and refining the empirical calculation of M_j are our future works.

3.4 Request scheduling and chunk transmission

Now, we introduce the strategy that schedules users requests to the right agent devices to fetch requested chunks. For each agent device, it serves all received requests with best efforts. For each user, the file downloading is conducted 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 agent devices. After completing the download of the first chunk, the request for the second chunk will be sent out. For each chunk request, addresses of 200 agent devices chosen randomly with the requested chunk will be returned to the user. The user will contact these agent devices to download content simultaneously.

For data transmission, each chunk is further divided into blocks with size 2MB for each block. However, some chunks may not be divisible by 2MB. For some chunks with 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 a chunk simultaneously from multiple agent devices.

4. MEASUREMENT AND PERFORMANCE

After introducing system architecture and discussing some detailed strategies, we present measurement study in this section.

4.1 Data collection

In current Thunder Crystal system, there are more than 11K active agent devices. A data report module is embedded into the software installed on agent devices, which reports necessary information to a log server farm. Reported information includes the number of agent devices that are working at any moment, measured instant uploading speed, event-driven messages recording the access log of content and other activities.

We collect records from more than 11K agent devices over one month period. Fat agents take about 25%. These agent devices are located over all geographical regions in China and cover almost all ISPs in China. We totally collect 35 billion reports with the size around 3TB. Hadoop platform is used to process this dataset for further analysis.

4.2 Resource distribution

In Thunder Crystal, each agent needs to contribute two kinds of resources, i.e., upload bandwidth and storage. We measure each device's upload capacity and storage capacity and present the cumulative distribution function (CDF) of each kind of resource in Fig. 2.

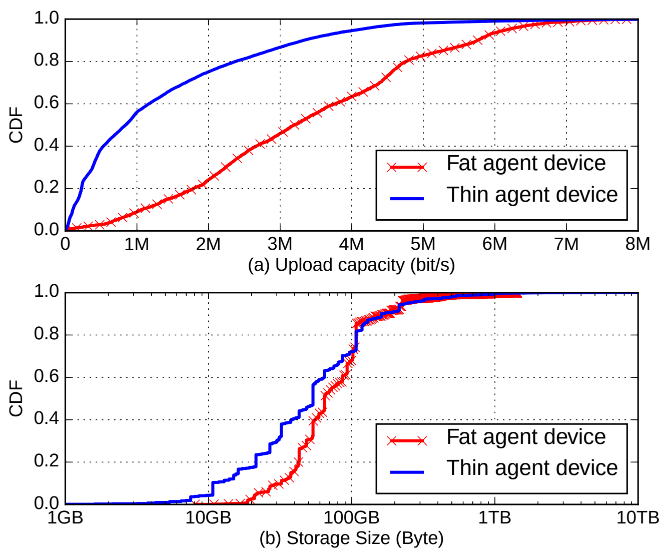

Figure 2: CDF of upload capacity and storage size.

It is not easy to get the knowledge of a device's upload capacity, so we use an approximated method to estimate upload capacity by recording each device's peak upload speed on daily basis. For each device, total 30 peak upload speeds are recorded in one month period. The median peak speed is taken as the device's upload capacity. As shown in Fig. 2(a), around 60% thin agent devices have less than 1Mb/s upload capacity, and around 80% fat agent devices have no more than 5Mb/s upload capacity. In general, the fat agents are equipped with much more powerful network resource.

The storage capacity can be easily measured and collected. CDF curve of agent devices' storage capacity is plotted in Fig. 2(b). We find that most agent devices allocate from 10GB to 100GB storage for content caching and fat agents contribute more storage than thin agents. However, a considerable number of fat agents use less storage than thin agents. Interestingly, the devices with extremely large storage (more than 20TB) are from thin agents with upload capacity less than 2Mb/s. Based on the observations above, we can infer that some agents use unreasonable storage resource for caching content.

4.3 Agent behavior

Agent behavior is very essential to understand how the system works. Our behavior study focuses on the agent device's online time. Device's online time reflects how well the agents are motivated to join in the game and how stable their service is.

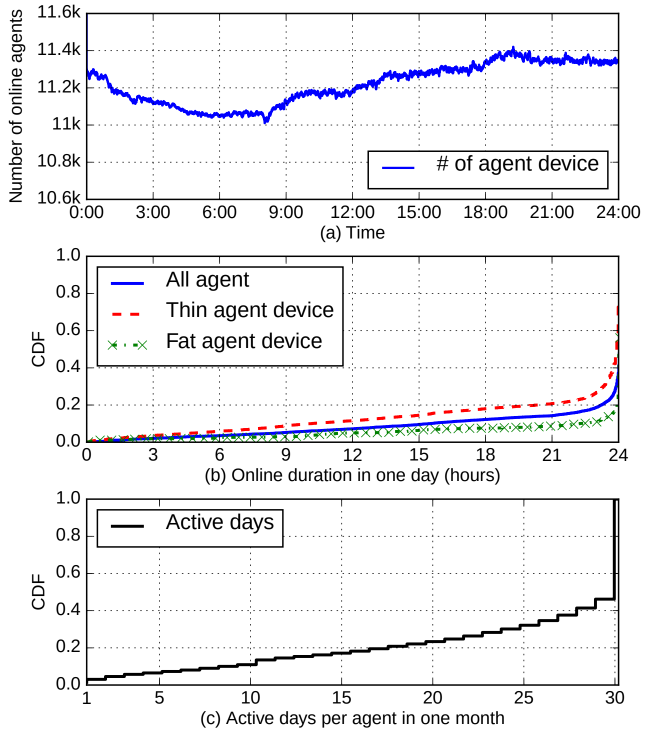

Figure 3: Agents behavior observation.

Fig. 3(a) shows the varying number of active devices during a single day. The number of active devices is between 11K and 11.4K, which is a rather small range. Fig. 3(b) plots the CDF curve of devices' online duration time during one day. Most agents keep online the whole day and fat agents' devices have longer online time. Fig. 3(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% agents are active every day during the measured one month time. Based on the above observations, we can conclude that Thunder Crystal can provide quite stable content distribution service since most agents have strong motivation to keep online. Agent devices act as mini CDN servers instead of P2P peers.

4.4 Dynamics of content popularity

Thunder Crystal is mainly used to serve video distribution. As reported in previous work [4], the distribution of video popularity is very skewed. In addition, video popularity decays very quickly with time. Thus, it is very essential to study the dynamics of content popularity in Thunder Crystal.

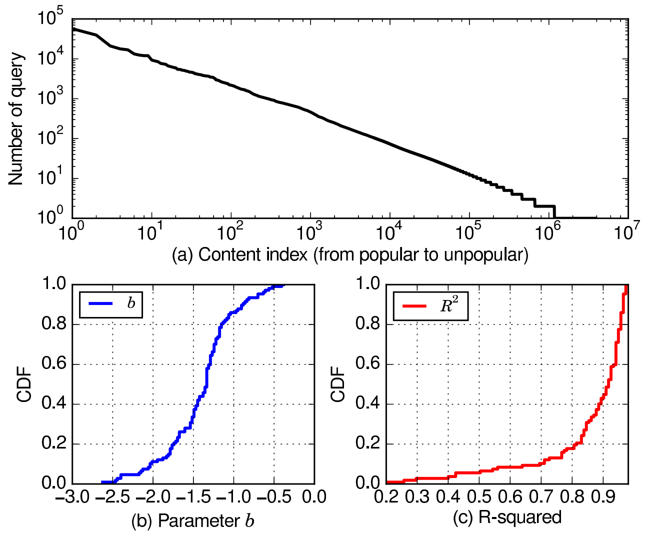

Figure 4: Dynamic of content popularity.

Through data collection, we know the number of requests for each file in each day. Percentage of the number of requests received by a particular file is taken as the file's popularity. Then, we can plot the popularity distribution of all files, as shown in Fig. 4(a). Content popularity follows power law distribution, which is consistent with related work [4], .

In order to study how the file popularity evolves with time, we use $f(x) = ax^b$ to fit the trace of the request number of each file once it is pushed into Thunder Crystal over 30 days. Parameter b reflects how fast the file popularity decays with time. CDF curves of all files' b and R-squared results are plotted in Fig. 4(b) and Fig. 4(c). As we can see, b is less than -1 for most files, implying fast popularity decay rate.

In addition to popularity dynamics, popularity skewness also affects system performance. For Thunder servers, only 80TB traffic budget can be used to push content to agent devices each day. To be efficient, Thunder servers should push out popular files. If the distribution of popularity is skewed, i.e., most users focus on fetching fewer files, then less files but more replicas of each file will be pushed out. Otherwise, more files but fewer replicas of each file will be pushed out. Fig. 5 illustrates this by comparing the content pushing results of day A with more skewed popularity distribution and day B with less skewed popularity distribution. Top 749 files on day A and top 2242 files on day B are pushed by Thunder servers.

4.5 System efficiency

We evaluate the Thunder Crystal's efficiency from two aspects: bandwidth utilization and amplification of Thunder

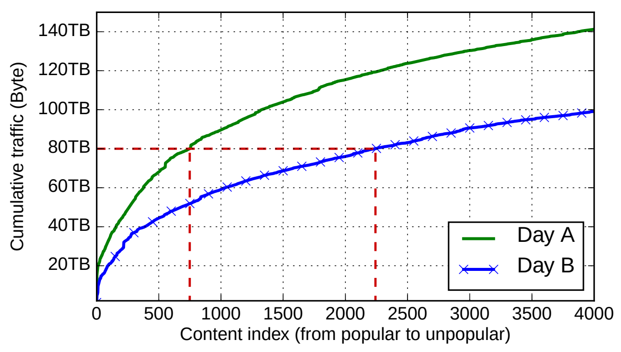

Figure 5: Cumulative traffic in the content deployment.

servers' upload traffic. Given upload capacity of device i is C_i, the maximum upload capacity of the whole system is $\sum_i C_i$. However, due to various reasons, e.g., the absence of requested content, system's peak bandwidth C_S is much less than the maximum upload capacity. We define system efficiency $\eta = \frac{C_s}{\sum_i C_i}$ to measure how efficiently bandwidth is utilized.

Thunder servers push total 80TB traffic to agent devices per day. How much is the total traffic from all agents reflects how efficiently the bandwidth is utilized. Therefore, we define amplification of upload traffic as $\gamma = \frac{\sum T_i}{80TB}$, where T_i is device i's upload traffic.

The daily values of η and γ over 30 days are plotted in Fig. 6(a) and Fig. 6(b). As we can see, the efficiency of bandwidth utilization is just between 0.5 and 0.7, implying more space for improvement. Allocating larger storage could be a method to improve η and we will further discuss this issue in the next subsection. Total Traffic is amplified by 10$\sim$20 times, which shows the significant achievement gained by Thunder Crystal.

4.6 Agent efficiency

Since Thunder Crystal's efficiency is determined by each agent's efficiency, it is also necessary to design metrics to evaluate the efficiency of each agent. We implement this study also from two aspects: bandwidth utilization and storage utilization.

Similar to the definition of η, we define bandwidth utilization as $\eta_i = \frac{u_i}{C_i}$ for device i, where u_i is device i's average upload speed, as the metrics to evaluate how efficiently a device's bandwidth is utilized. The storage utilization of a device refers to the used storage divided by the device's total available storage. CDF curves of both bandwidth utilization and storage utilization are plotted in Fig. 7. As we can see, thin agent devices, most probably with smaller download capacity, have less storage utilization but higher bandwidth utilization efficiency. This means that storage resource of fat agents is not allocated enough such that their bandwidth resource is not utilized as efficiently as thin agents.

In order to further explore the relationship between storage utilization and bandwidth utilization, we plot the values of all devices, each as a point, in Fig. 8 with x axis representing storage size and y axis representing upload traffic. Surprisingly, we find a strong positive correlation (about 0.515) between upload traffic and used storage. That means fat agents with larger upload capacity should allocate more storage resource to achieve higher bandwidth utilization.

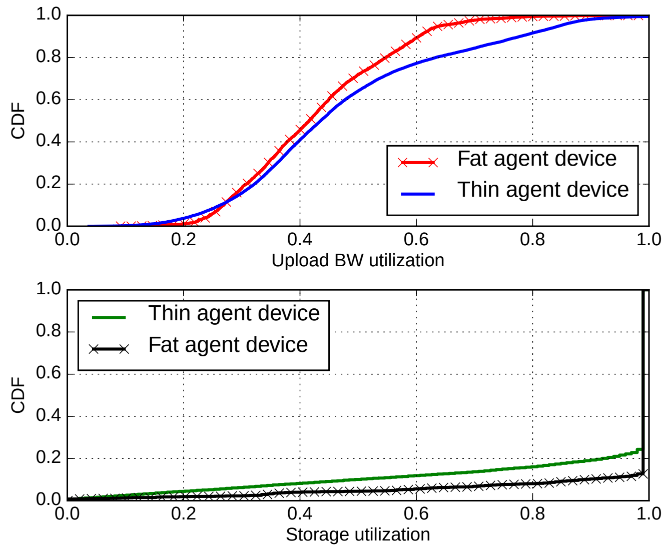

Figure 7: Upload bandwidth utilization and storage utilization for fat and thin agents.

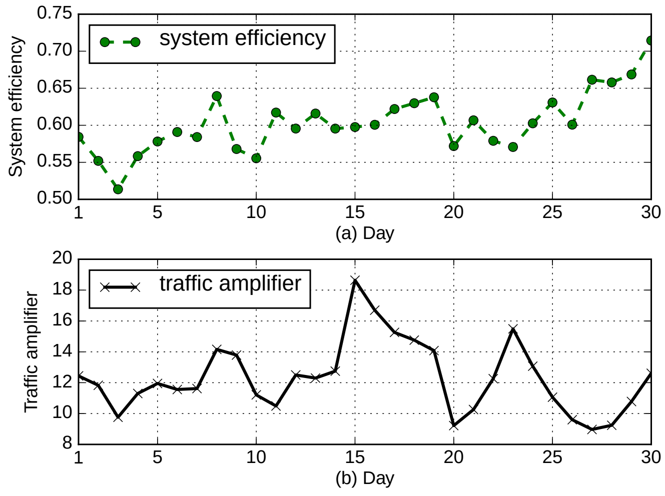

Figure 6: System efficiency and traffic amplifier in one month.

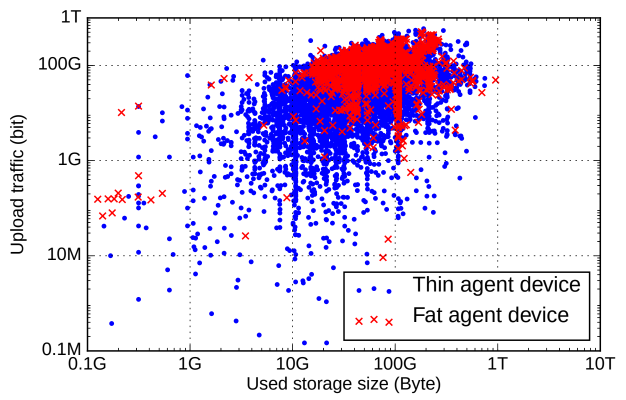

Figure 8: Scatter plot of used storage size and upload traffic.

5. RELATED WORK

To accelerate content distribution in large-scale networks, there are mainly CDN, P2P and hybrid CDN/P2P approaches on the market. CDN systems resort to dedicated edge servers to ensure quality of service [5]. The system performance can be improved by developing both scheduling and replica caching strategies [7, 11]. Compared with pure CDN systems, P2P VoD systems are much more cost efficient, and have been studied from theoretical analysis [16, 10] to practical system design and evaluations [8, 6].

Hybrid CDN/P2P approach [14] is proposed to obtain scalability and low cost advantages of P2P along with the reliability and manageability of CDNs. Cost reduction is a major concern in such systems [3]. However, simply combining P2P with CDN can not effectively address all P2P weaknesses, e.g., unstable online time, scarce storage for replication, criticism for possible leakage of content copyright and so on. Thunder Crystal provides a crowdsourcing-based content delivery service so as to augment both advantages of CDN and P2P.

It is a challenge to motivate users to devote upload bandwidth in traditional P2P networks because of users' inelastic streaming demand. In [12], Wu et al. designed an incentive mechanism that rewards each peer based on its dedicated upload bandwidth. [15] provided a general framework to analyze various incentive protocols. Yang et al. [13] proposed a profitable business model to analyze the benefits for content service providers, ISPs and end users in a competing market. Different from the schemes above, the Thunder Crystal system [2] encourages users to contribute bandwidth and storage resource by rebating cash, and reduces cost to serve ICPs.

6. CONCLUSION

In this work, we introduce the design principles and key strategies of novel crowdsourcing based content distribution platform, Thunder Crystal. By rebating cash to agents based on their upload traffic, agents are well motivated to contribute their resources. By offloading the content tasks to tens of thousands agents, Thunder Crystal not only achieves cheaper bandwidth, but also amplifies its content distribution capacity. Finally, we propose the metrics to evaluate system efficiency and agent efficiency. We also present measurement study to show how the system works.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback that helped improve this paper. The research was partially supported by the Natural Science Foundation of China (grant no. 61402297), the Natural Science Foundation of SZU (grant no. 201437). We acknowledge the Foundation of Shenzhen City (grant no. KQCX20140519103756206).

7. REFERENCES

[1] Xiaomi mi wifi wireless ap router. `http://www.xiaomishop.com/128-original-xiaomi-mi-wifi-wireless-router.html`.

[2] Xunlei shuijing. `http://shuijing.xunlei.com/`.

[3] A. Balachandran, V. Sekar, A. Akella, and S. Seshan. Analyzing the potential benefits of cdn augmentation strategies for internet video workloads. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 43–56. ACM, 2013.

[4] L. Chen, Y. Zhou, and D. M. Chiu. A lifetime model of online video popularity. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–8. IEEE, 2014.

[5] C. Huang, A. Wang, J. Li, and K. W. Ross. Measuring and evaluating large-scale cdns. In *ACM IMC*, volume 8, 2008.

[6] Y. Huang, T. Z. Fu, D.-M. Chiu, J. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p-vod system. In *ACM SIGCOMM computer communication review*, volume 38, pages 375–388. ACM, 2008.

[7] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize cdn performance. In *Proceedings of the 9th ACM SIGCOMM IMC*, pages 190–201. ACM, 2009.

[8] Z. Liu, C. Wu, B. Li, and S. Zhao. Uusee: large-scale operational on-demand streaming with random network coding. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[9] A. Passarella. A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Computer Communications*, 35(1):1–32, 2012.

[10] B. Tan and L. Massoulié. Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM Transactions on Networking (TON)*, 21(2):566–579, 2013.

[11] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM CCR*, 43(5):5–12, 2013.

[12] W. Wu, R. T. Ma, and J. C. Lui. Distributed caching via rewarding: An incentive scheme design in p2p-vod systems. *Parallel and Distributed Systems, IEEE Transactions on*, 25(3):612–621, 2014.

[13] L. Yang and W. Lou. 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*, page 33. IEEE Press, 2012.

[14] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 25–34. ACM, 2009.

[15] B. Q. Zhao, J. C. Lui, and D.-M. Chiu. A mathematical framework for analyzing adaptive incentive protocols in p2p networks. *Networking, IEEE/ACM Transactions on*, 20(2):367–380, 2012.

[16] Y. Zhou, T. Z. J. Fu, and D. M. Chiu. Statistical modeling and analysis of p2p replication to support vod service. In *Proceedings of IEEE INFOCOM*, pages 945–953, 2011.