Joint Request Balancing and Content Aggregation in Crowdsourced CDN

Ming Ma[1], Zhi Wang[2], Kun Yi[1], Jiangchuan Liu[3], and Lifeng Sun[1]

[1]Department of Computer Science and Technology, Tsinghua University
[2]Graduate School at Shenzhen, Tsinghua University
[3]College of Natural Resources and Environment, South China Agricultural University
{mm13@mails., wangzhi@sz., yik13@}tsinghua.edu.cn
csljc@ieee.org, sunlf@tsinghua.edu.cn

Abstract—**Recent years have witnessed a new content delivery paradigm named crowdsourced CDN, in which devices deployed at edge network can prefetch contents and provide content delivery service. Crowdsourced CDN offers high-quality experience to end-users by reducing their content access latency and alleviates the load of network backbone by making use of network and storage resources at millions of edge devices. In such paradigm, redirecting content requests to proper devices is critical for user experience. The uniqueness of request redirection in such crowdsourced CDN lies that: on one hand, the bandwidth capacity of the crowdsourced CDN devices is limit, hence devices located at a crowded place can be easily overwhelmed when serving nearby user requests; on the other hand, contents requested in one device can be significantly different from another one, making request redirection strategies used in conventional CDNs which only aim to balance request loads ineffective. In this paper, we explore request redirection strategies that take both workload balance of devices and content requested by users into consideration. Our contributions are as follows. First, we conduct measurement studies, coving 1.8M users watching 0.4M videos, to understand request patterns in crowdsourced CDN. We observe that the loads of nearby devices can be very different and the contents requested at nearby devices can also be significantly different. These observations lead to our design for request balancing at nearby devices. Second, we formulate the request redirection problem by taking both the content access latency and the content replication cost into consideration, and propose a request balancing and content aggregation solution. Finally, we evaluate the performance of our design using trace-driven simulations, and observe our scheme outperforms the traditional strategy in terms of many metrics, e.g., we observe a content access latency reduction by 50% over traditional mechanisms such as the Nearest/Random request routing scheme.**

I. Introduction

Video content traffic has shown tremendous growth over the past several years, and it has already become the largest Internet traffic category [1], [2]. To improve the content access quality experienced by users as well as alleviate the content delivery load for both CDN servers and backbone network, recently, Content Delivery Networks (CDNs) have been making use of the network and storage resources of devices in the edge network (even at people's homes) to assist their video service [3], [4], [5], [6], [7]. As such edge network based content delivery platform acts as a crowdsourcing system which offloads content distribution tasks to massive edge devices, we name it after crowdsourced CDN and refer to the edge devices as content hotspots (or hotspots). In crowdsourced CDN, CDN servers keep pushing content to edge hotspots; while edge hotspots are encouraged to contribute their storage to cache content and upload bandwidth to distribute content to end users. For example, Youku, one of the most popular online video providers in China, has deployed over 300K smart Wi-Fi Access Points (APs) as the content hotspots to cache and distribute videos [8]. Due to the short distance between content hotspots and user requests, the transmission of videos which are prefetched to hotspots can be done very efficiently, i.e., reducing the content access latency experienced by end users.

In the crowdsourced CDN, redirecting content requests to proper hotspots is critical for user' experience. The uniqueness of request redirection in such crowdsourced CDN lies that: on one hand, the bandwidth capacity of the edge content hotspots is tightly limited (apparently smaller than CDN servers). When hotspots serve nearby users, their workloads are directly influenced by the neighbouring user density distribution. Hence the workload of each individual hotspot will be highly volatile and many hotspots can be easily overwhelmed while others are under-utilized; on the other hand, contents requested in the hotspots can be significantly different from each other due to the influence of small population [9] (i.e., the popularity of contents requested at each hotspot is influenced by a small number of local requests), making request redirection strategies used in conventional CDNs only aiming to balance request loads ineffective. Meanwhile, the scalability is another important problem because request redirection strategies have to be efficiently enforced to massive individual hotspots.

Previous studies on request redirection can be divided into two categories. (i) *Load balance based solutions* [10], [11]: which assume the content is replicated on all CDN sites. They redirect requests to different CDN sites based on latency, server load, and the traffic cost. The limitation of such solutions for crowdsourced CDN is that the content popularity in crowdsourced CDN is significantly affected by local users [9], making the request redirection inefficient without considering the content placement. (ii) *Content-based solutions* [12], [3], [6], which jointly consider the user redirection and content placement problem based on content locality, bandwidth costs, and storage capacity. But they assume the CDN servers have enough capacity to serve the requests. This assumption is unrealistic for the edge hotspots in crowdsourced CDN, making the load balance be a fatal factor for request redirection.

In this paper, we propose a request redirection strategy that takes both hotspots' load balance and content requested by users into consideration. Our contributions are as follows.

▷ First, we carry out large-scale measurement studies on mobile video request patterns and today's deployment of crowdsourced CDN infrastructure. We use the following datasets: a video session dataset containing 1.8M users watching 0.4M videos in 59M sessions, provided by a major video provider in China; and a Wi-Fi Access Points (APs) dataset containing the deployment information of 1M APs which can be used as content hotspots in crowdsourced CDN. In our trace-driven experiments, we select the Wi-Fi APs as the content hotspots and redirect requests to hotspots based on their geo-location. Our measurement insights are as follows: (1) Workloads of nearby hotspots can be very diverse, making cross-hotspot collaboration for the request balancing promising; (2) Contents requested at nearby hotspots can be significantly different, making content an important factor in request redirection. Our comprehensive measurement studies not only identify the request balancing problem in crowdsourced CDN, but also provide the invaluable guidelines for our solution design.

▷ Second, based on our measurement insights, we formulate the request redirection and content replication problem in crowdsourced video CDN. We propose a Request-Balancing and Content-Aggregation (RBCAer) algorithm, a novel request redirection solution that not only achieves load balance across content hotspots but also minimizes content replication costs for the content provider. Firstly we transform the request balancing as a minimum-cost-maximum-flow (MCMF) network problem; Then we incorporate *content aggregation* into the framework, in which we guide the workload redirection between hotspots with high content similarity, to alleviate the content replication costs incurred by the load balance.

▷ Third, we design trace-driven experiments to verify the effectiveness and efficiency of our solution. Through extensive trace-driven evaluation, we show that our design outperforms conventional request redirection solutions including nearest routing scheme and random routing scheme, e.g., compared to Nearest/Random scheme, RBCAer can improve the hotspot serving ratio, reduce the content access latency by 42% and reduce the CDN server load by 20%.

The rest of the paper is organized as follows. Sec II conducts measurement to motivate our design. Sec III introduces the system model and problem definition. Sec IV gives a detailed description of our algorithm, and Sec V presents the simulation results. Sec VI reviews related works. A conclusion is drawn in Sec VII.

II. MEASUREMENT AND MOTIVATION

We illustrate the proposed system architecture in Fig. 1. There is a scheduling server which learns the global system information and makes the user redirection and content replication decision.

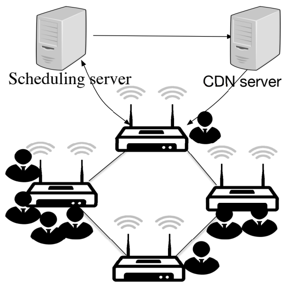


Fig. 1: The architecture of crowdsourced CDN.

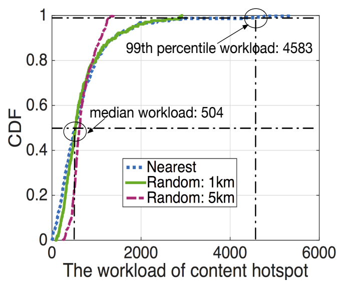


Fig. 2: Workload distribution of content hotspots.

Firstly, we use a measurement study to motivate our work. In this section, we 1) quantify the high skewness of the hotspot load distribution in crowdsourced CDN; 2) demonstrate the collaborative potential among content hotspots from the perspective of load diversity and content similarity. We then highlight the importance of considering the cost constraint when optimizing the user routing in crowdsourced CDN for VoD streaming.

Firstly, we introduce two datasets used for our crowdsourced CDN measurement:

▷ **Video session traces** collected by iQiyi[1], one of the most popular online video providers in China. The traces were collected in 2 weeks of May 2015, in Beijing, containing 1.8M users watching 0.4M videos in 59M sessions. Each trace item records four fields, including the global unique user identifier, the video session timestamp, the requested video title, and the location where the user watches the video (reported by the video player based on built-in GPS function);

▷ **Wi-Fi traces** provided by a famous Internet company in China. The dataset contains the location for over 1M Wi-Fi APs in Beijing, which occupies a large fraction of Wi-Fi APs that are actually deployed in Beijing.

Note that the network hop distance[2] may be more suitable for us to evaluate latency for user routing. However, due to the difficulty of collecting such statistics, we assume that network latency between two devices in the network is proportional to their geo-distance [13].

To conduct the following measurement, we need to sample some Wi-Fi APs as the content hotspots for crowdsourced CDN. There are 300K content hotspots in the latest news of Youku, and the population in Beijing accounts for 1.5% of the population in China[3], so we randomly sample $300\text{K}\cdot 1.5\% = 5\text{K}$ hotspots from the Wi-Fi AP datasets of Beijing for our measurement (unless otherwise specified).

A. Two Strategies to Reveal the System Design Tradeoff

Quantifying the load difference in the Nearest Routing strategy. In the crowdsourced CDN, each content hotspot naturally has the incentive to serve the video requests nearby to minimize the network access latency, and this is a typical

[1]iQiyi website: http://www.iqiyi.com/
[2]Network hop definition: https://en.wikipedia.org/wiki/Hop_(networking)
[3]Beijing population: http://www.bjstats.gov.cn/zt/rkjd/sdjd/201603/t20160322_340767.html

solution for the crowdsourced CDN. We map each request to its nearest hotspot through the geo-location information in the datasets. The line with legend “Nearest” in Fig. 2 shows the workload distribution of hotspots using the video request trace of 1 day (other days are similar). Specifically, we observe that the 99th-percentile workload can be up to $9\times$ of the median, and this high skewness is caused by the density of mobile users’ geo-distribution.

Due to the significant load difference among hotspots, some hotspots may be overloaded while others may be under-utilized. It results in the inefficiency for crowdsourced CDN. *Since the overloaded hotspots which violate the capacity constraints impair the system effectiveness, we need a request balancing solution among hotspots that can address the load variance in crowdsourced CDN.*

Quantifying the highly content replication cost for the Random Routing Strategy Since the skewed workload in the Nearest Routing Strategy leads to a significant reject ratio, a straightforward way to balance the load of hotspots is randomly redirecting a request to any of the hotspots within a certain of distance threshold, i.e., Random Routing Strategy. To identify the effectiveness of the random scheme, we set the 1km and 5km as the Random Routing threshold. The measurement results show that permitting users to leverage distant hotspots, i.e., at the cost of higher latency, is indeed an effective approach that can reduce the workload variance and increase the hotspot serving ratio observed from Fig. 2.

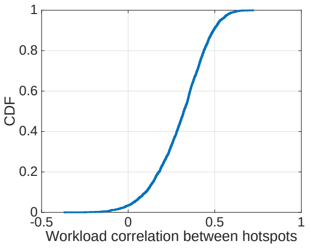
To explore the content replication cost, we assume every requested content should be replicated to the hotspots. By calculating the sum number of requested contents in each content hotspot, we observe the content replication cost for the Random Routing Strategy with threshold 1km (resp. 5km) is 10% (resp. 23%) more than the content replication cost in the Nearest Routing Strategy. It indicates content hotspot caches additional contents to serve users distant from it. However, it is unreasonable that hotspots pre-fetch contents as much as possible for serving more requests in crowdsourced CDN, because a large amount of content replication cost increases the CDN server load and the cache size of hotspots is finite compared to CDN server (even if it is not limited in our measurement).

In summary, *although we can route users to a different hotspot and make hotspots cooperate with each other to improve the hotspot serving ratio, the concrete scheme for user redirection should be well designed to avoid inefficient replication.*

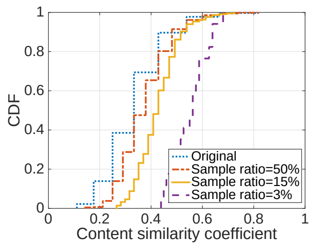
B. Cooperation Potential and Design Insights

We examine two design spaces in the hotspot coordination, including workload correlation and content similarity correlation between pairs of hotspots.

Workload correlation among hotspots. We calculate the workload correlation (under the Nearest Routing Strategy) over timeslots (1h) during 1 day between hotspots. Fig. 3(a) shows the CDF of Spearman correlation between any two hotspots with distance lower than 5km. There are 70% pairs of hotspots having the correlation under the 0.4, indicating a low-level dependency between pairs of nearby hotspots. *Hence hotspots could cooperate with each other spanning the time of day for load balancing*, e.g, peak video delivery demand in residential districts may be at night while another place like a company may be have low demand at the night.



(a) Workload correlation.



(b) Content similarity coefficient.

Fig. 3: Cooperation potential among content hotspots.

Cache Cooperation among Hotspots. We calculate the content correlation between hotspots under the Nearest Routing Strategy. To identify the content similarity with a different number of hotspots in the system, we randomly sample 50%, 15% and 3% of the original 5K hotspots as 3 new content hotspot sets. For each content hotspot set, we calculate the Jaccard similarity coefficient of the two sets of Top-20%[4] contents for any pair of hotspots using Eq. 1, i.e., the size of the intersection divided by the size of the union on two content sets $\mathcal{V}_i$ and $\mathcal{V}_j$.

$$Jaccard(\mathcal{V}_i,\mathcal{V}_j) = \frac{\|\mathcal{V}_i \cap \mathcal{V}_j\|}{\|\mathcal{V}_i \cup \mathcal{V}_j\|}. \tag{1}$$

As shown in Fig. 3(b), we plot the CDFs of the Jaccard similarity coefficient between any two hotspots whose distance between them is lower than 5km. Considering a hotspot covers a smaller region in case of more sampled hotspots, we observe the popular contents are less similar between smaller regions. We make the line labelled “Original” demonstrate the content similarity between hotspots which cover small regions in the crowdsourced CDN, and the line labelled “Sample ratio=3%” demonstrate the content similarity between servers which cover a large region in the traditional CDN. We observe the content similarity between two nearby hotspots can be very diverse ranging from 0.1 to 0.8 in the crowdsourced CDN, which is different from the CDN servers. *Therefore, the traditional request redirection strategy which only considers the server load and ignores the content distribution is not suitable for the crowdsourced CDN, hence we should carefully design the request redirection scheme by making use of the content similarity between hotspots in crowdsourced CDN.*

III. System Model and Problem Formulation

In this section, we describe the system model and define the request redirection problem considering both of the content access latency and content placement cost.

[4]We choose Top-20% videos due to the video popularity follows the 80/20 rule of the Pareto principle: https://en.wikipedia.org/wiki/Pareto_principle

In our system, we make the following assumptions for the formulation convenience:

$\triangleright$ The connection betweens video requests and content hotspots do not change during the video session, and one request is only served by one hotspot or original CDN server.

$\triangleright$ As the number of user requests is too massive (compared with the number of content hotspots) for us to make redirection decision individually, they are aggregated to their nearest hotspots and we redirect these incoming requests among hotspots to achieve the load balance.

$\triangleright$ We assume that all of the videos reside permanently at the CDN server. Besides, each video has an identical size 1 and each user request has unit demands 1 (if not, the original video can be divided into fixed-size segments).

$\triangleright$ The popularity distribution of the files changes slowly [14], and it can be learned through some popularity prediction algorithm (like the regression model ARIMA [15]). After the content placement is determined, the hotspots prefetching contents to their caches for the video requests.

A. System Model

In reality, a crowdsourced CDN provider owns M content hotspots $\mathcal{H} = \{1, 2, ..., M\}$ at fixed locations in the network. The content hotspots are co-located at Wi-Fi APs and all of them are connected with each other via the backhaul network. We denote the service (resp. storage) capacity for each hotspots h as s_h (resp. c_h) in terms of the number of requests can be handled by it in one timeslot (resp. the number of contents it can cache). These content hotspots operate in conjunction with the original CDN server and cache the popular videos based on the incoming requests, chosen from a set of O videos $\mathcal{V} = \{1, 2, ..., O\}$. For ease of presentation, the size of each video in $\mathcal{V}$ is one unit. This is rational because videos can be split into equal-sized chunks.

In each timeslot t, we assume that a set of N user requests $\mathcal{R} = \{1, 2, ..., N\}$ can be served by any of the M content hotspots. And we denote the map function between the user request r and the video requested v as $W(r) = v, v \in \mathcal{V}$. Users will firstly issue the content request to the scheduling server. If the requested video k which is presented in the suitable content hotspots, then the request is scheduled to be served immediately. However, if content v is not presented in the suitable hotspots, the scheduling server then forwards the request to the original CDN peering server. Let d_{rh} denote the latency incurred by redirecting request r to hotspot h and CDN server, respectively.

Intuitively there are two binary decision variables in the system:

- The redirection between user requests and hotspots is denoted as a binary $N \times M$ matrix $\mathcal{X}$, and variable x_{ij} indicates whether the user request i is redirected to hotspot j $x_{ij} = 1$ or not $x_{ij} = 0$. If no hotspot is available to serve request i, it would be redirected to the original CDN server S, i.e, $x_{iS} = 1$.
- The assignment between contents and hotspots is denoted as a binary $O \times M$ matrix $\mathcal{Y}$, and variable y_{kj} indicates whether the video v_k is assigned to hotspot j $y_{kj} = 1$ or not $y_{kj} = 0$. And we assume the original CDN server caches all of the videos.

B. Problem Definition

We consider a user redirection problem with the consideration of minimizing the content access latency and the content replication cost:

$\triangleright$ Accumulated content access latency. The most crucial target of implementing the crowdsourced CDN is to shorten the distances between content hotspots and user requesting, and thereby make the use experience low content access latency. This factor can be calculated as below:

$$\Omega_1 = \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{H} \cup \{S\}} x_{ij} d_{ij}. \tag{2}$$

$\triangleright$ Content replication cost. We note that there are a substantial amount of prior works on user redirection and content placement algorithms for edge content delivery, see e.g. [3], [16], [4], [6] and references therein. To the best of our knowledge, all these related works neglect the content replication cost. But in crowdsourced video CDN, an abundant of content hotspots are scheduled to cache content, which incurs influential pressure for the CDN server in the crowdsourced CDN[7], [17]. Thus the content replication cost is a significant factor as below:

$$\Omega_2 = \sum_{j \in \mathcal{H}} \sum_{k \in \mathcal{O}} y_{kj}. \tag{3}$$

Then this joint request redirection and content replication problem can be formulated as the following optimization function (U):

$$\min_{\mathcal{X}, \mathcal{Y}} \alpha \Omega_1 + \beta \Omega_2, \tag{U}$$

s.t.

$$\sum_{j \in \mathcal{H}} x_{ij} + x_{iS} = 1, \tag{4}$$

$$x_{ij} \le y_{W(i)j}, \qquad \forall i, j, \tag{5}$$

$$\sum_{i \in \mathcal{R}} x_{ij} \le s_j, \tag{6}$$

$$\sum_{k \in \mathcal{V}} y_{kj} \le c_j. \tag{7}$$

Where α, β are weight factors to tune the objective function. Eq. 4 ensures that each request must be served by a content hotspot or original CDN server. Eq. 5 ensures that if a request i is assigned to hotspot j, then i's requested video $W(i)$ must be placed on the hotspot j. Eq. 6 and Eq. 7 are needed to achieve the service capacity and storage capacity.

Clearly, the problem (U) is very hard to solve optimally. [4] proves that the joint user redirection and content placement problem (JUR-CP) which only minimizes the content access latency is polynomial-time reducible to the NP-hard problem—Unsplittable Hard-Capacitated Metric Facility Location Problem (UHCMFL). And it is easy to see that solving an instance of the problem (U) is equivalent to solving an instance of the JUR-CP problem in [4]. Thus problem (U) is also NP-hard. Hence we should design an efficient solution for the crowdsourced video CDN.

C. Formulation Transformation

Recall that we aggregate each user request to its nearest hotspot for the simplification, hence in each timeslot t, we denote λ_h as the number requests aggregated to the hotspot $h \in \mathcal{H}$, and λ_{hv} as the number requests for video v that are aggregated to h, and $\lambda_h = \sum_{v \in \mathcal{V}} \lambda_{hv}$. Since we denote CDN server as S, $\lambda_S = 0$. As the aggregated requests at different hotspots can be significantly different, some hotspots may be overloaded while others may be under-utilized. We assume that all content hotspots are reachable from each other, then a hotspot can redirect a fraction of its video request to another hotspot. Denote f_{ij} and f_{iS} as the number of requests redirected from hotspot i to hotspot j ($i \neq j$) and CDN server S respectively. In terms of the flow characteristics in the flow network, There are the following constraints on f_{ij}:

$$f_{ij} = \begin{cases} -f_{ji}, & if \quad i \neq j \\ 0, & if \quad i = j \end{cases} \qquad \forall i, j \in \mathcal{H} \cup \{S\} \tag{8}$$

$$\sum_{i \in \mathcal{H} \cup \{S\}} \sum_{j \in \mathcal{H} \cup \{S\}} f_{ij} = 0, \tag{9}$$

$$\sum_{j \in \mathcal{H} \cup \{S\}} max\{f_{ij}, 0\} \leq \lambda_i, \qquad \forall i \in \mathcal{H} \cup \{S\}, \tag{10}$$

$$\lambda_i - \sum_{j \in \mathcal{H} \cup \{S\}} max\{f_{ij}, 0\} + \sum_{j \in \mathcal{H} \cup \{S\}} max\{f_{ji}, 0\} \leq s_i, \forall i \in \mathcal{H}. \tag{11}$$

Eq. 8 ensures that for any two hotspot i and j, the flow of user request in terms from hotspots i to hotspots j is the negative of the flow from hotspots j to hotspots i. As the requests from any given hotspot i to itself is zero, we have $f_{ii} = 0$. Eq. 9 ensures that all flow is conserved. Finally, Eq. 10 ensures that the sum of all outgoing request from hotspot i is not greater than its incoming request λ_i, and Eq. 11 ensures that the final request flows served by hotspot i is not greater than its service capacity s_i.

Then we rewrite the Eq. 2 as follows:

$$\theta = \sum_{i \in \mathcal{H} \cup \{S\}} \sum_{j \in \mathcal{H} \cup \{S\}} d_{ij} \cdot max\{f_{ij}, 0\} \tag{12}$$

Using this flow network based formulation, we present our heuristic workload scheduling mechanism in the next section.

IV. Request Balancing and Content Aggregation for Crowdsourced CDN

In practice, as the optimization is performed over massive and widespread users and content hotspots, scalability is a crucial issue. In this section, we present a polynomial time algorithm, i.e., Request Balancing and Content Aggregation algorithm (referred to as RBCAer), a novel workload scheduling mechanism that leverages the workload, latency and content distribution in different hotspots to wisely schedule user requests. Firstly, we consider the request balancing as a minimum-cost-maximum-flow (MCMF) network problem; Secondly, we incorporate *content aggregation* into the framework to decrease the content replication costs incurred by the request balancing among hotspots.

A. Request Balancing Flow Network Model

Our workload scheduling mechanism makes decisions at the beginning of each time slot (e.g., 1h). We start our algorithm with a Load-Balance-aware MCMF model used in RBCAer. To minimize the accumulated latency, firstly we consider requests should be served by its nearest hotspots as much as possible. Hence we redirect extra requests from overloaded hotspots to under-utilized hotspots by considering the latency as the cost in MCMF model. We denote the set of overloaded hotspots as $\mathcal{H}_s = \{i | i \in \mathcal{H}, \lambda_i > s_i\}$ and set of under-utilized hotspots as $\mathcal{H}_t = \{i | i \in \mathcal{H}, \lambda_i < s_i\}$. We determine the upper bound of the outgoing or incoming flows ϕ_i for each hotspot i based on its service capacity s_i, i.e., $\phi_i = |s_i - \lambda_i|$. The total workload that can be offloaded from $\mathcal{H}_s$ to $\mathcal{H}_t$ is $maxflow = \min\{\sum_{i \in \mathcal{H}_s} \phi_i, \sum_{j \in \mathcal{H}_t} \phi_j\}$.

Then we need to determine the value of f_{ij}, i.e., the number of requests that should be redirected from hotspot i to hotspot j for all $i \in \mathcal{H}_s$ and $j \in \mathcal{H}_t$, to minimize the accumulated network latency. We construct a flow network $G_d = (V, E)$ in Fig. 4(a), where the vertex set V contains the set of overloaded hotspots $\mathcal{H}_s$ and the set of under-utilized hotspots $\mathcal{H}_t$, together with a source vertex $source$ and a sink vertex $sink$. For each edge between the $source$ and overloaded hotspots $i \in \mathcal{H}_s$, the link capacity is set to ϕ_i, and the cost is set to 0. Similarly, we set the capacity and cost of the edge between $j \in \mathcal{H}_t$ and $sink$ to ϕ_j and 0 respectively.

For adding edges between $i \in \mathcal{H}_s$ and $j \in \mathcal{H}_t$, we give a threshold θ and only add edge $< i, j >$ when $d_{ij} < \theta$ to achieve graph simplification and algorithm efficiency below. For each edge $< i, j >$ between $\mathcal{H}_s$ and $\mathcal{H}_t$, its service capacity is set to $\phi_{ij} = \min\{\phi_i, \phi_j\}$, and their cost is set to delay d_{ij}. The final network flow graph $G_d(V, E)$ is illustrated in Fig. 4(a).

Given a network graph $G_d(V, E)$, we could determine the f_{ij} by finding the maximum flow in $G_d(V, E)$ from $source$ to $sink$ such that the total latency cost of the flow is minimized. However, the graph $G_d(V, E)$ fails to capture the content-related factor for the request redirection. Thus in the next subsection, we modify the $G_d(V, E)$ to a flow graph which also cares for the content replication cost.

B. Content Aggregation Flow Network Model

In order to reduce the content replication cost incurred by the request redirection from the overloaded hotspots to under-utilized hotspot, we update $G_d(V, E)$ to a content aggregation one $G_c(V, E)$.

Our basic idea is to greedily urge a set of overloaded hotspots with higher content similarity to offload their requests to one under-utilized hotspots. Firstly, using the hierarchical clustering algorithm [18], we cluster the hotspots based on

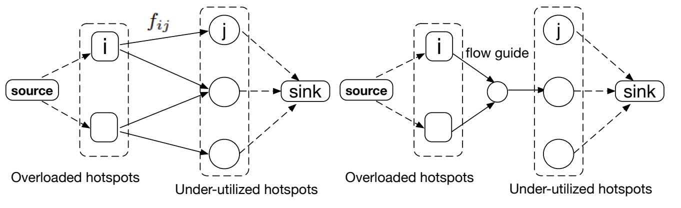


Fig. 4: The algorithm framework: (a) Request balancing in MCMF network $G_d(V,E)$; (b) Content aggregation flow network $G_c(V,E)$.

the content-aware distance between pairs of hotspots which is calculated as follows:

$$J_d(i,j) = 1 - Jaccard(V_i, V_j), \tag{13}$$

where $Jaccard(V_i, V_j)$ is shown in Eq. 1. We restrict the distance $J_d(i,j)$ between any two hotspots in the same cluster lower than 0.5 (it is suitable for our hotspot clustering). We denote the clustering results as K cluster of hotspots $\mathcal{P} = \{P_1, P_2, ...P_K\}$. $P_k \in \mathcal{P}$ is the set of hotspots belongs to cluster P_i, and any two items in $\mathcal{P}$ are disjoint.

Next, we construct $G_c(V,E)$ illustrated in Fig. 4(b) following the hotspot cluster. Initially, we have $G_c(V,E) = G_d(V,E)$. In order to guide the overloaded hotspots in a cluster to offload their workloads to the under-utilized hotspot in the same cluster, we insert the flow guide nodes about the hotspot cluster to the $G_c(V,E)$. We denote function $SourcetoSink(\cdot)$ (resp. $SinktoSource(\cdot)$) to map the overloaded hotspot i (under-utilized hotspot j) to the set of under-utilized hotspots (resp. overloaded hotspots) which connect with i (resp. j), i.e., $SourcetoSink(i) = \{j| <i,j> \in E\}$ (resp. $SinktoSource(j) = \{i| <i,j> \in E\}$). We denote $\mathcal{H}_{jk} = \{i | i \in SinktoSource(j), i \in P_k\}$.

For each under-utilized hotspot j and each hotspot cluster P_k, if $\sum_{i \in \mathcal{H}_{jk}} \phi_{ij} \geq 1/2\phi_j$ or overloaded hotspots in $\mathcal{H}_{jk}$ belongs to the cluster of hotspot j, we add a flow guide node n_{kj} to $G_c(V,E)$. This setting could guarantee this guide node is influential for urging hotspot $i \in \mathcal{H}_{jk}$ offloads their workloads to hotspot j. For flow guide node n_{kj}, we modify the edges in $G_c(V,E)$ as follows: 1) Add edges between overloaded hotspot $i \in \mathcal{H}_{jk}$ and flow guide node n_{kj} with capacity ϕ_{ij} and cost 0. 2) Add edges between the flow guide node n_{kj} and under-utilized hotspot j with capacity ϕ_{ij} and cost $\frac{\sum_{i \in \mathcal{H}_{jk}} \phi_{ij}}{\|\mathcal{H}_{jk}\|}$; 3) Remove edges $<i,j>$ $(i \in \mathcal{H}_{jk})$. So far we obtain the network flow graph $G_c(N,E)$ shown in Fig. 4(b) derived from Fig. 4(a).

Next, we demonstrate our complete algorithm using the flow graph $G_c(N,E)$.

C. Content Aggregation and Request Balancing Algorithm

Algorithm 1 shows the detail of our RBCAer algorithm. In the initialization (from line 1 to line 4), we calculate the total value of moveable workloads $maxflow$ in line 4. We aim at finding a feasible solution f_{ij} for achieving the $maxflow$. After the initialization, it iteratively computes partial f^*_{ij} with the latency threshold θ (line 6), which starts at θ_1, until θ is larger than the θ_2 or a feasible solution f_{ij} is found (line 5).

The tuning parameters θ_1, θ_2 and δ are to balance the content-similarity influence and the cost of latency. Recall in Sec. IV-A, we can guarantee all edge cost in $G_d(V,E)$ is lower θ. The low value of θ implies few edges between $\mathcal{H}_s$ and $\mathcal{H}_t$, which results in fewer moveable workloads between $\mathcal{H}_s$ and $\mathcal{H}_t$ in $G_c(V,E)$ and consequently reduces the influence of content-similarity-aware flow network. The crowdsourced CDN providers could tune θ_1, θ_2 and δ_d in accordance with specific conditions.

After the iteration, we examine whether a feasible solution f_{ij} is found. If not, the value of f^*_{ij} for the remaining moveable workloads is calculated based on $G_d(V,E)$ (line 12) or redirected them to CDN server (line 14).

Next subsection details how we determine the specific video request redirection and the content placement strategy according to f_{ij}.

```
Algorithm 1: RBCAer Algorithm.
Input  : The set of video request $\mathcal{R}$; the service capacity $s_i$
         and storage size $c_i$ for each hotspot $i \in \mathcal{H}$; the
         network latency matrix $d_{ij}$ $(i,j \in \mathcal{H})$; and tuning
         parameters $\theta_1$, $\theta_2$, $\delta_d$.
Output: Inter-hotspot flow redirection $f_{ij}$ between
         overloaded and under-utilized hotspots, and video
         replication decision $y_{vj}$ $(\forall v \in \mathcal{V}, j \in \mathcal{H})$.
1  $f_{ij} \leftarrow 0$;      $\theta = \theta_1$;
2  Construct the graph $G_d(V,E)$ with $\theta$;
3  Construct the graph $G_c(V,E)$ based on $G_d(V,E)$;
4  $maxflow \leftarrow \min\{\sum_{i \in \mathcal{H}_s} \phi_i, \sum_{j \in \mathcal{H}_t} \phi_j\}$;
5  while $\theta <= \theta_2$ and $\sum_{i \in \mathcal{H}_s} \sum_{j \in \mathcal{H}_t} f_{ij} < maxflow$ do
6      Calculate ${f_{ij}}^*$ in $G_c(V,E)$ by invoking MCMF
       algorithms in [19];
7      $f_{ij} \leftarrow f_{ij} + {f_{ij}}^*$;      $\phi_i \leftarrow \phi_i - {f_{ij}}^*$;
8      $\phi_j \leftarrow \phi_j + {f_{ij}}^*$;      $\theta \leftarrow \theta + \delta_d$;
9      Update $G_d(V,E)$, $G_c(V,E)$ with current $\phi_i$ and $\theta$;
10 end
11 if $maxflow - \sum_{i \in \mathcal{H}_s} \sum_{j \in \mathcal{H}_t} f_{ij} > 0$ then
12     Calculate ${f_{ij}}^*$ for residuary unmoved workload:
       update $G_d(V,E)$ with current $\phi_i$ and $\theta$, and invoke
       MCMF algorithms in [19];
13     $f_{ij} \leftarrow f_{ij} + {f_{ij}}^*$;
14     For the unmoved workloads which cannot be handled
       within $\theta_2$, redirect them to original CDN server;
15 end
16 Calculate $y_{vj}$ by invoking Procedure 1
   ContentAggregationReplication($f_{ij}$);
17 return $f_{ij}, y_{vj}$;
```

D. Content Aggregation Procedure

According to the flow redirection result f_{ij}, we conduct the request redirection and content replication. The details are illustrated in Procedure 1. To redirect requests with similar content set from overloaded hotspots to similar under-utilized hotspots, We design 3 efficiency indexes: 1) The redirecting efficiency index $e_f(i,v,j)$ is video v's request volume which could be redirected from hotspot i to j (line 2); 2) The content placement efficiency index $e_u(v,j)$ is the sum of the video v' request volume which could be redirected from overloaded hotspot $i \in SinktoSource(j)$ to j, i.e., $e_u(v,j) \leftarrow \sum_{i \in SinktoSource(j)} e_f(i,v,j)$ (line 4); And 3) the offload efficiency index $e_l(v,i)$ is the video v's request volume in hotspot i (line 5).

For the goal of reducing the content replication cost incurred by the offloading from the overloaded hotspots to the under-utilized hotspots, we determine which videos' requests should be redirected from which overloaded hotspots to which under-utilized hotspots based on the sorted content placement efficiency $e_u(v,j)$. The redirection continues until we find a feasible solution to achieve f_{ij}. For the final content replication, we make the replication continue until all of the caches are filled (line 16) or server load reaches the peak traffic B_{peak} observed (line 15). At the end, RBCAer returns the content placement decision y_{vj}.

We use the classical Ford-Fulkerson algorithm [19] to solve the MCMF problem, and it is the most time-consuming process in our algorithm. Hence the time complexity of RBCAer is $O(|V|^2|E|)$ time complexity, where $|V|$ is the number of hotspots in $G_c(V,E)$. $|E|$ is the number of edges in $G_c(V,E)$ whose maximum number is on $|V|^2$ order of magnitudes. In the next section of the algorithm evaluation, we will show our θ used in the graph generation could greatly reduce the value of $|E|$ which benefits the algorithm scalability and efficiency.

Procedure 1: ContentAggregationReplication

```
Input  : The redirected video request $f_{ij}$ between $\mathcal{H}_s$ and
         $\mathcal{H}_t$; the storage size $c_i$ for $i \in \mathcal{H}$; the value of
         requests $\lambda_i$ for $i \in \mathcal{H}$; the value of requests $\lambda_{iv}$ for
         video $v$ in $i \in \mathcal{H}$.
Output: Video placement solution $y_{vj}$, for all $v \in \mathcal{V}, j \in \mathcal{H}$.
1  $\mathcal{L}_f \leftarrow \mathcal{H}_s \times V^{(T)} \times \mathcal{H}_t$, $e_f(i,v,j) \in \mathcal{L}_f$;
2  $e_f(i,v,j) \leftarrow min\{f_{ij}, \lambda_{vi}\}$;
3  $\mathcal{L}_u \leftarrow \mathcal{V}^{(T)} \times \mathcal{H}_t$;
4  $e_u(v,i) \in \mathcal{L}_u$;
   $e_u(v,j) \leftarrow \sum_{i \in SinktoSource(j)} e_f(i,v,j)$;
5  $\mathcal{L}_l \leftarrow \mathcal{V}^{(T)} \times \mathcal{H}$; $e_l(v,i) \in \mathcal{L}_l$;    $e_l(v,i) \leftarrow \lambda_{vi}$;
6  if $\lambda_{vi} > 0$, $y_{vi} = 1$;
7  Rank $\mathcal{L}_u$ in the descending order of $e_u(v,j)$;
8  do
9      Select $(v', j') \in \mathcal{L}_u$ with the largest placement
       efficiency index;
10     For all $i' \in SinktoSource(j')$, redirect the
       $e_f(i',v',j')$ requests of video $v'$ from hotspot $i'$ to
       $j'$, and $f_{i'j'} \leftarrow f_{i'j'} - e_f(i',v',j')$; Update $y_{vj}$
11     $e_l(v',i') \leftarrow e_l(v',i') - e_f(i',v',j')$;
12     $e_l(v',j') \leftarrow$
       $e_l(v',j') + \sum_{i' \in SinktoSource(j')} e_f(i',v',j')$;
       $f_{ij} \leftarrow f_{ij} -$
13 while for all $f_{ij} = 0,\ i \in \mathcal{H}_s,\ j \in \mathcal{H}_t$
14 Rank $\mathcal{L}_l$ in the descending order of $e_{v,i}$;
15 while $B_{cur} < B_{peak}$ and $\mathcal{L}_l \neq \Phi$ do
16     Select $(v', i') \in \mathcal{L}$ with the largest offload efficiency
       index and $c^*_{i'} < c_{i'}$;
17     $c^*_{i'} \leftarrow c^*_{i'} + 1$; Replicate content $v'$ to hotspot $i'$;
18     $B_{cur} \leftarrow B_{cur} + 1$ and $\mathcal{L}_l \leftarrow \mathcal{L}_l - \{(v', i')\}$;
19 end
20 return $y_{vj}$
```

V. Performance Evaluation

In this section, we conduct an extensive trace-driven evaluation and present the numerical results. Specifically, using the real-world dataset, we characterize the effectiveness offered by our proposed algorithms RBCAer, and compare it with other conventional request redirection schemes according to the performance and running time.

A. Simulation Setup and Methodology

To simplify and speed up our simulation, we choose a rectangular region (17km $\times$ 11km) in Beijing from our datasets including a rich collection of $212,472$ requests, $15,190$ videos and 310 content hotspots. Fig. 5 shows the geo-distribution of these requests and content hotspots. Unless otherwise specified, we consider the collaboration between overloaded hotspots and under-utilized hotspots within a circular shape with radius 1.5km in default, i.e., $\theta_1 = 0.5$km, $\theta_2 = 1.5$km, and $\delta_d = 0.5$km. And each hotspot $i \in \mathcal{H}$ is endowed with a cache size equalling 3% of the entire video set size (i.e., $c_i = 450$), and its service capacity s_i suffices for transmitting 5% of the entire video set (i.e., $s_i = 760$).

We mainly compare RBCAer to the following frequently used schemes in crowdsourced video CDN:

▷ Nearest scheme: when a request is generated, it is routed to the nearest hotspot. Accordingly, each hotspot caches the most popular files based on the requests of the nearby users independently from the others.

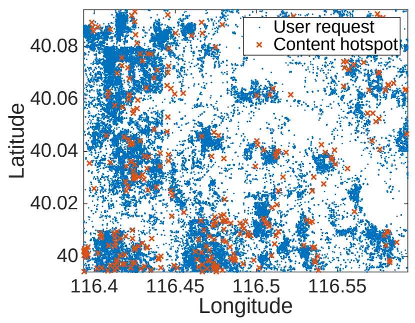


Fig. 5: The geo-distribution of video requests and content hotspots.

▷ Local random scheme (referred to as Random scheme) [5], [7]: the content hotspot serves users and caches the most popular videos within a certain radius nearby (1.5km in our simulation). When a request is generated, it is randomly routed to a hotspot within 1.5km that has stored a copy of the associated video.

1) Performance Metrics: We illustrate four system performance metrics in the evaluation and their normalization methods used in the figure illustration as follows:

▷ Hotspot serving ratio, i.e., after prefetching contents to edge hotspots, the proportion of requests that can be served by the hotspots (normalized by original CDN workload, i.e., the total number of requests).

▷ Average content access latency/distance. Recall we let the distance between requests and hotspots reflect the latency. Since the maximum distance between pairs of hotspots in our dataset is 20km ($i.e., \sqrt{17^2 + 11^2}$km), we directly set the content access latency as 20km when a user request is served by CDN server.

▷ Content replication cost: Prefetching contents to hotspots introduces extra load to the original CDN server. It is the total number of replicas that are prefetched to the hotspot (normalized by the entire video set size.)

▷ CDN server load, i.e., the original CDN workload minus the number of requests served by the hotspots, and then plus the number of content replicating. It is normalized by the original CDN workload. It can reflect the overall system performance since it considers both of the content access condition and content replication cost.

B. Overall Performance: Parameter Impact Analysis

1) The Impact of the Service Capacities: We analyze the impact of the hotspots' service capacities on the algorithm performance in Fig. 6. We vary the capacity per hotspots from 2% to 5% of the entire video set size in Fig. 6. As expected, in Fig. 6(a), increasing service capacity could increase the hotspot serving ratio. We observe that: 1) The hotspots will be in overloaded conditions when they have low service capacity. Hence simply prefetching the s_i most popular video for hotspot is suffices to fully utilize the service capacity, and consequently the gaps among these three schemes are small; 2) The performance gap between RBCAer and other two schemes increases as capacity increases, e.g., in Fig. 6(a), to achieve the hotspot serving ratio of 0.74, the hotspots only need service capacity of 4% by using our RBCAer, while hotspots need more capacity which equals 5.2% (resp. 5.7%) by using the Random scheme (resp. Nearest scheme); 3) The RBCAer scheme consistently outperforms the Nearest and Random schemes, and the serving ratio gap increases in the range of 0% ($s_i = 2\%$) to 12% ($s_i = 7\%$).

Two crucial factors which play an important role in our RBCAer are content access distance and replication cost demonstrated in Fig. 6(b) and Fig. 6(c). From Fig. 6(b), we observe our RBCAer provides significant content access distance reduction by carefully design the request balancing solution, e.g., when capacity value equals 5%, the content access distance of RBCAer is 42% lower than that of Nearest and Random schmes (from 6km to 3.5km). In Fig. 6(c), we observe Random scheme and Nearest scheme are almost unchangeable and mainly restrained by the hotspot cache size, e.g., Random scheme prefetches most video ($4.54\times$ of entire video size) since hotspots not only prefetch their local popular videos but also popular videos of other location. Our content placement cost is the lowest and changeable in response to various capacity. This is because that in RBCAer, 1) our content aggregation procedure would migrate and aggregate a popular video from multiple overloaded hotspots to one under-utilized hotspot, and it reduces the content placement cost; and 2) various capacity influences the size of overloaded hotspot set and under-utilized hotspot set, i.e., too low capacity (resp. too high capacity) brings less under-utilized hotspots (resp. overloaded hotspots) to serve the extra requests (resp. offload their extra requests), and consequently it reduces the improvement space for content aggregation procedures in RBCAer.

Finally, Fig. 6(d) shows the CDN server load after using these three schemes, and RBCAer significantly outperforms the Nearest and Random schemes. When capacity value equals 5%, the CDN server load reduces to 47% of its original load with our RBCAer, which is about 22% lower than the Nearest and Random schemes (i.e., $(60\% - 47\%)/60\% = 22\%$).

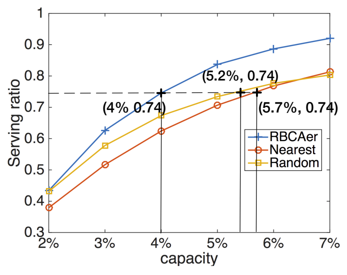
2) The Impact of the Cache Sizes: Fig. 7 compares the performance of the discussed schemes with different cache size of hotspots. Parameter c_i varies from 0.5% to 5% of the entire video set size (note the ticks in x-axis is uneven). We observe our RBCAer obviously outperforms both Nearest and Random schemes. The details are analyzed as follows:

1) Increasing the available cache size improves the hotspots serving ratio for all the strategies, as more videos are cached at the hotspots. Our RBCAer has the best performance, e.g., as shown in Fig. 7(a), RBCAer solution could achieve 0.7 of the hotspot serving ratio with only 0.67% of the cache size, while the Random scheme (resp. Nearest scheme) requires more available cache space, i.e., 2% (resp. 3%) of the entire video size;

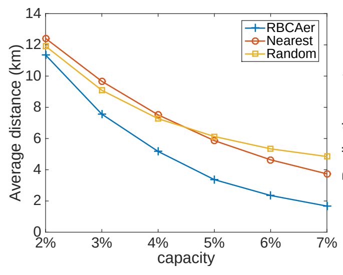
2) Fig. 6(b) shows the average content access distance by using RBCAer is obviously below the other two schemes under all of the cache size setting. Specifically, RBCAer can achieve about 50% of the content access distance reduction over Nearest and Random schemes.

3) The content replication cost quickly increases as cache size increases. Since the sharp changes make the differences among the three strategies hard to observe, we zoom in two details at cache size 0.9% and 3% respectively. At the cache point of 0.9%, the content replication cost of RBCAer is tiny (0.01% of the entire video set size) higher than the Nearest Strategy. This is because that, to achieve the request balancing, RBCAer deploys extra contents for under-utilized hotspots to serve the requests offloaded from the overloaded hotspots. The situation becomes different at the point of cache size 3% which is detailed in Fig. 6(c).

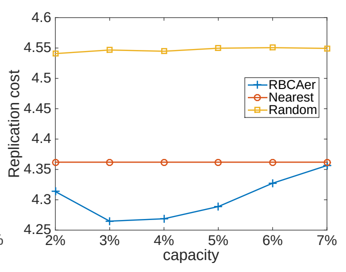
4) As expected, Fig. 6(d) shows that, increasing the available cache size, the CDN server load declines gradually at first and then increases gradually, since the replication cost grows faster than the number of requests served by hotspots when cache size higher than 1% of the entire video set size. When cache size equals 1%, our RBCAer solution outperforms Nearest and



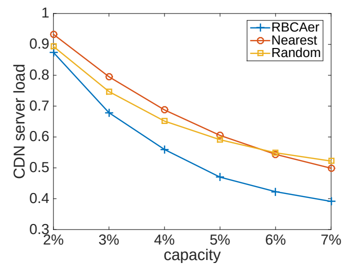
(a) Hotspot serving ratio.



(b) Average redirection distance.

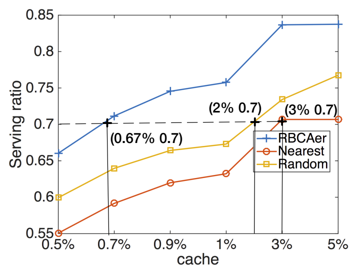


(c) Content replication cost.

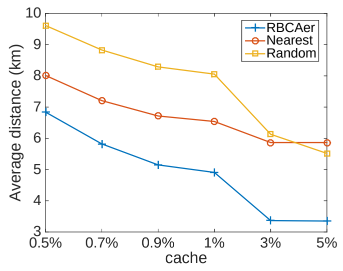


(d) CDN server workload.

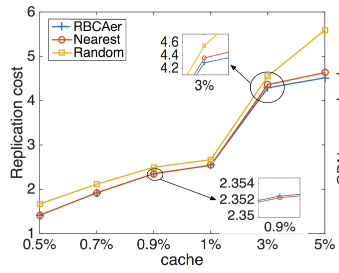
Fig. 6: Performance comparison for various capacity.



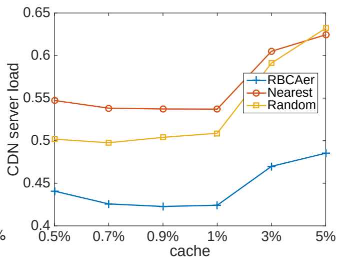
(a) Hotspot serving ratio.



(b) Average redirection distance.



(c) Content replication cost.



(d) CDN server workload.

Fig. 7: Performance comparison for various cache size.

Random schemes by reducing the CDN server load to 0.425 of the original server load, which 21% (resp. 17%) lower than the server load using Nearest scheme (resp. Random schemes).

C. Impact of Factor θ

Recall in Sec. IV-A we only generate edges whose cost lower than θ for graph G_d. Fig. 9 shows the influence of θ. We change θ from 0km to 7.5km, and calculate the number of edges in the G_d (normalized by the number of most possible edges $|V|^2$) and the maximum flow of $G_d(V, E)$ (normalized by the maximum flow when we add all edges between overloaded hotspots and under-utilized hotspots to $G_d(V, E)$, i.e., $maxflow$ in Sec. IV-A), respectively. We observe when θ is set to 7.5km, G_d could successfully achieve the $maxflow$ and the number of edges is only 11% of the $|V|^2$. In our evaluation, θ_2 is set to 1.5km which could handle about 50% of the $maxflow$. Considering the number of edges in the flow graph is an important factor influencing the time complexity to solve MCMF, hence in practical, restricting the hotspot cooperation within a nearby region is enough to balance hotspots' load and speed up the scheduling decision of the RBCAer.

D. Running Time

We leverage four different algorithms, including straightforward solving the linear relaxation of the integer programming formulation of the URBCA problem (referred to LP-based scheme), our RBCAer, Nearest scheme and Random scheme, to make the scheduling decision. We set up these solutions on a laptop which has 128GB memory, 2.6GHz GenuineIntel CPU. As Linear Programming (LP) relaxation and rounding scheme[5] (e.g., [4]) is a common technology to approximately solving the integer linear programming, we straightforward solve the LP relaxation of our ILP to evaluate the running time of such kind of algorithms. Due to our laptop's memory size, we sample 10K requests for the running of LP-based scheme by GLPK. For other three scheme, we still use our original dataset.

We repeat the experiments and compute the average running time. As presented in Fig. 8, perform the LP-based scheme (not to mention straightforward the ILP) consumes more than 2.4 hour, which is too computationally demanding to implement and not feasible in a real system. We find that RBCAer costs more computation time than Nearest and Random schemes, but consuming 35s to obtain the scheduling results is feasible for a real world crowdsourced CDN. And RBCAer reduces up to 20% CDN server load compared with Nearest/Random schemes. This result indicates that RBCAer is efficiency to wisely schedule a large-scale of request balancing and content replication for crowdsourced video CDN.

VI. Related Work

Video traffic already represents a significant fraction of today's network traffic, e.g., a recent report indicates that Netflix and Youtube account for 55% of the peak downstream traffic in North America [1]. To alleviate such video delivery workloads on backbone and CDN servers, in recent few years, both academia and industrial communities realize that the abundant of edge devices, such as set-top [16], small cell base station (SBS) [4], and smart Wi-Fi APs [7], can be well utilized as edge content hotspots, which prefetch and

[5]LP relaxation and rounding: The first step requires solving the LP relaxation of the ILP problem, and then rounding it to obtain a g - close integer solution

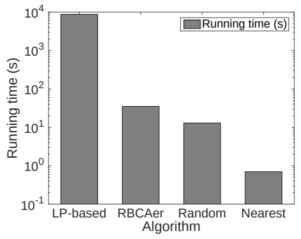


Fig. 8: Efficiency comparison of different scheduling algorithms.

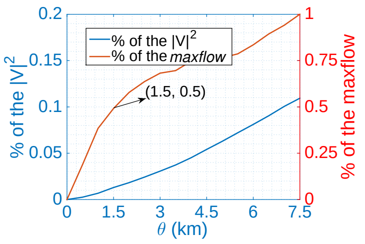


Fig. 9: The influence of θ.

deliver contents to nearby end users, to offload the traffic of CDN servers as well as improve the content access quality experienced by users. In this paper, we denote such platform as Crowdsourced video CDN.

In this section, we survey the related works on user redirection and content caching. Several past research efforts have considered these problems separately.

User redirection for load balance. Traditional content providers, e.g., Akamai replicates contents at essentially all CDN servers where demand exists, and balances the workload spanning different CDN providers [10], [11] and different time zone. In the peer CDN system, [20], [5] design distributed-hash-table (DHT) to guarantee the load balance by randomly selecting peers for each video request. These request redirection strategies are based on latency and server load. However, in crowdsourced CDN, the cache size of the content hotspots is limited and it is impossible to only think traffic engineering problem for the user routing. In this paper, we take the content placement cost into account and only hotspots with similar content similarity do the workload offloading cooperation.

Cooperative caching. In the conventional CDN, [21] develops a polynomial-time optimal algorithm for the hierarchical caching problem based on a reduction to MCMF model. [22] develops approximation caching algorithms aiming to minimize the bandwidth cost. [23] solve the caching problem based on the matching in a bipartite graph. [17] propose a dynamic content allocation for Cloud-assisted CDN to minimize the cloud traffic cost. However, these caching works assume caches' service capacity is never the bottleneck and hence do not consider the user redirection balancing problem.

Jointly request redirection and caching. A method to further increase the system benefits is to jointly design caching and routing policies. The measurement analysis in [24] indicates a great interaction between content distribution and traffic engineering could benefit the system performance. Similar to our design, [25] solves the joint content placement and request routing problem based on the min-cost network model and heuristic workload aggregation. But it is not in the context of video distribution and ignores the content placement cost. [26], [16], [27] employ the Lagrangian relaxation method and use iterative algorithms to reach a solution. However, there is no guarantee about the running time of the algorithms. Recently, caching and routing in small cell networks, with the objective of minimizing content access delay, has been studied in [3], [6], [4] without considering the content replication cost. [4] formulates this joint problem as UHCMFL Problem, which is most similar to our problem. But it solves the problem based on the LP-relaxation which requires a long running time for practical implementation. In our previous work [28], we present a region partition solution for the hotspot organization based on the latency and content replication cost. Based on the result of [28], if we aggregate all hotspots in each region to a virtual hotspot, RBCAer could be used to make cross-region cooperation to further increase the algorithm scalability.

Finally, it is worth observing that caches are located as close to the users creating a degree of similarity with P2P networks [16], [29]. Most works on P2P algorithms however pay attention to minimizing the content delivery traffic for a given content placement, whereas our focus is on a more controlled environment where the request balancing and content replication can be actively and centrally managed.

To the best of our knowledge, the user redirection problem both considering the content placement cost and request balancing has not been well addressed in the crowdsourced CDN. Our proposed RBCAer solution can effectively conduct balanced user redirection without much content replication cost and with polynomial-time running time.

VII. Conclusions

In this paper, we explore, formulate and address the request redirection problem in crowdsourced CDNs by taking both hotspots' load balance and content requested by users into consideration. Specifically, we conduct measurement studies to understand request patterns in crowdsourced CDN, finding that the workloads of nearby hotspots are very different, and contents requested at nearby hotspots can also be significantly different. Based on these observations, we then formulate the user redirection problem by taking both the load balance and the content requested by users into consideration. To address this problem, we propose a heuristic scheduling solution, RBCAer, which obtains the scheduling decision by solving a content aggregation network flow problem (designed by the content similarity) to achieve request balancing and content aggregation. Finally, we evaluate the effectiveness and efficiency of RBCAer using trace-driven simulations in terms of many system performance metrics and running time.

Acknowledgement

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61402247, 61472204, and 61521002, Beijing Key Laboratory of Networked Multimedia under Grant No. Z161100005016051, and SZSTI under Grant No. CXZZ20150323151850088.

References

[1] S. report, "Global internet phenomena–2015 latin america and north america report."
[2] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update 2015-2020 white paper."
[3] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *INFOCOM*. IEEE, 2012.

[4] K. Poularakis, G. Iosifidis, and L. Tassiulas, “Approximation algorithms for mobile data caching in small cell networks,” *ToC*, 2014.

[5] L. Chen, Y. Zhou, M. Jing, and R. T. Ma, “Thunder crystal: a novel crowdsourcing-based content distribution platform,” in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2015, pp. 43–48.

[6] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, “On the complexity of optimal routing and content caching in heterogeneous networks,” in *INFOCOM*. IEEE, 2015.

[7] M. Ma, Z. Wang, K. Su, and L. Sun, “Understanding content placement strategies in smartrouter-based peer video cdn,” in *NOSSDAV*. ACM, 2016.

[8] “http://www.prnewswire.com/news-releases/chinacache-and-youku-router—-creating-a-new-cdn-ecosystem-300138554.html.”

[9] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, “Placing dynamic content in caches with small population,” *INFOCOM*, 2016.

[10] H. Xu and B. Li, “Joint request mapping and response routing for geo-distributed cloud services,” in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 854–862.

[11] S. Narayana, J. W. Jiang, J. Rexford, and M. Chiang, “To coordinate or not to coordinate? wide-area traffic management for data centers,” *Princeton Univ., Princeton, NJ, USA, Tech. Rep*, 2012.

[12] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, “Volley: Automated data placement for geo-distributed cloud services.” in *NSDI*, vol. 10, 2010, pp. 28–0.

[13] O. Krajsa and L. Fojtova, “Rtt measurement and its dependence on the real geographical distance,” in *TSP, 2011 34th International Conference on*. IEEE.

[14] Z. Li, J. Lin, M.-I. Akodjenou, G. Xie, M. A. Kaafar, Y. Jin, and G. Peng, “Watching videos from everywhere: a study of the pptv mobile vod system,” in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 185–198.

[15] G. P. Zhang, “Time series forecasting using a hybrid arima and neural network model,” *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[16] W. Jiang, S. Ioannidis, L. Massoulié, and F. Picconi, “Orchestrating massively distributed cdns,” in *CoNEXT ’12*. ACM.

[17] G. Dán and N. Carlsson, “Dynamic content allocation for cloud-assisted service of periodic workloads,” in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 853–861.

[18] S. C. Johnson, “Hierarchical clustering schemes,” *Psychometrika*, 1967.

[19] D. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ, USA: Princeton University Press, 2010.

[20] Y. Xia, S. Chen, C. Cho, and V. Korgaonkar, “Algorithms and performance of load-balancing with multiple hash functions in massive content distribution,” *Computer networks*, vol. 53, no. 1, pp. 110–125, 2009.

[21] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, “Placement algorithms for hierarchical cooperative caching,” *Journal of Algorithms*, vol. 38, no. 1, pp. 260–302, 2001.

[22] S. Borst, V. Gupta, and A. Walid, “Distributed caching algorithms for content distribution networks,” in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[23] K. Poularakis and L. Tassiulas, “Optimal cooperative content placement algorithms in hierarchical cache topologies,” in *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*. IEEE, 2012, pp. 1–6.

[24] A. Sharma, A. Venkataramani, and R. K. Sitaraman, “Distributing content simplifies isp traffic engineering,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1. ACM, 2013, pp. 229–242.

[25] H. Qian and M. Rabinovich, “Application placement and demand distribution in a global elastic cloud: A unified approach.” in *ICAC*, 2013.

[26] T. Bektaş, J.-F. Cordeau, E. Erkut, and G. Laporte, “Exact algorithms for the joint object placement and request routing problem in content distribution networks,” *Computers & Operations Research*, 2008.

[27] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, “Collaborative hierarchical caching with dynamic request routing for massive content distribution,” in *INFOCOM, 2012 Proceedings IEEE*.

[28] W. Hu, Z. Wang, M. Ma, and L. Sun, “Edge video cdn: A wi-fi content hotspot solution,” *Journal of Computer Science and Technology (JCST)*, 2016, accepted for publication.

[29] Y. Huang, Z. Xias, Y. Chen, R. Jana, M. Rabinovich, and B. Wei, “When is p2p technology beneficial to iptv services,” *Proc. of NOSSDAV, IL, USA (June 2007)*, 2007.