Towards Problem of First Miss under Mobile Edge Caching

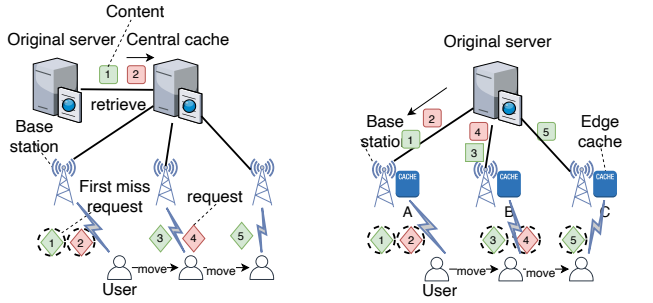
Yanpeng Luo*, Chao Song*, Haipeng Dai†, Zhaofu Chen‡, Nianbo Liu*, Ming Liu* and Jie Wu§
*School of Computer Science and Engineering, University of Electronic Science and Technology of China, CHINA
†State Key Laboratory for Novel Software Technology, Nanjing University, CHINA
‡Huawei, CHINA
§Center for Networked Computing, Temple University, USA
Email:{chaosong, csmliu}@uestc.edu.cn, haipengdai@nju.edu.cn, jiewu@temple.edu

***Abstract*—Mobile Edge Caching (MEC) can cache content at the edge of the network to reduce the delay and overhead of content transmission, which has become an effective method to solve the explosive growth of network traffic. To make good use of the limited resources in edge devices, many contents caching strategies use various methods to predict the popularity of content. However, caches get close to the edge of the network can lead to the rapid increase of caches' number and the user's requests are dispersed into a large number of caches, which leads to the popularity distribution of contents in edge caches is quite different and the number of first miss requests (the corresponding content is requested for the first time and is not in the cache) in edge caches becoming an essential factor affecting the cache hit rate. This paper first demonstrates the significant impact of the first miss requests through dataset analysis and establishes a mathematical model for the first miss problem in the edge cache. Then we analyze the similarity of requests received by caches and propose a proactive push algorithm based on similarity to improve the hit rate of edge caches. Through the trace-driven simulation experiment, we verify that the methods proposed in this paper can significantly improve the caches' hit rate.**

I. Introduction

Mobile Edge Caching (MEC) utilizes storage provided by mobile edge servers and is a use case of Mobile Edge Computing [1]. A cache-enabled mobile edge server can be an independent server attached to Base Stations (BSs) or User Equipment (UE). Proactive caching at the edge of the network has fully demonstrated its advantages in improving user experience and offloading traffic [2]. It avoids network congestion and reduces the delay of contents transmission by pushing contents to caches in advance. The previous works are mainly divided into two categories, one focus on analyzing and optimizing caching policy with known content popularity. Since the popularity of contents changes dynamically, other types of works are mainly to predict the popularity of content.

Most of the early work assumed that the popularity of contents, i.e., the probability of contents being requested, was known. C. Yang etl. in [3] advocate proactively caching popular contents when the network is off-peak. At the same time, the popularity of content is difficult to obtain, Y. Zhan etl. in [4] use neural networks to predict the popularity of contents. Since contents popularity changes fast, the dataset will quickly become out of date, S. Li etl. in [5] propose an online learning algorithm that learns the short-term popularity of content. When caches are close to users, the number of requests received by caches will decrease [6] [7] [8], which is difficult to make accurate predictions. By gathering requests from local caches, [8] propose an age-based algorithm, which solves the problem of small samples in the local caches. The key to all the above works is to obtain the popularity of contents. They all assume that some contents in caches have a higher probability of being requested than other contents. i.e., the popularity of contents in caches meets the Zipf distribution.



(a) Central cache. (b) Edge cache.

Fig. 1. The problem of first miss in central/edge cache: (a) count of first miss requests is 2, (b) count of first miss requests is 5.

When the popularity of contents on edge caches does not meet the Zipf distribution, the caches hit rate will be decreased. When new content is requested for the first time in an edge cache and causes a miss, we term this request as a *first miss request*. From the perspective of the entire network, the small number of contents has most of the traffic, and the popularity of the contents should conform to the Zipf distribution [6]. However, with the rapid growth of the number of caches, requests are dispersed into more caches, and the popularity of the contents in the cache no longer follows the Zipf distribution. The problem of first miss at edge cache is shown in Fig. 1(a), when there is only one central cache in the entire network, there will only be one first miss request for each content. When the cache is close to the edge of the network, shown in Fig. 1(b), due to the increase in the number of caches, the requests are directed to the nearest caches, and the first miss requests generated by each content will increase as the number of corresponding caches increases. These missed

requests will lead to cache retrieve content from the original server, resulting in high latency and network overhead. First miss requests did not attract enough attention because it is generally believed that there will be a large number of requests for some contents in caches, and the first miss requests only account for a small part of all requests. However, due to the limitation of coverage, the difference of contents popularity on edge caches is tiny, then the first miss requests in the edge caches will account for a large proportion of the requests, thus affecting the caches hit rate and causing a bad user experience. We call this problem as the problem of first miss at edge cache.

In this paper, we model the problem of first miss at edge cache and solve this problem by proactive pushing contents in advance according to the similarity between caches. We first analyzed that broadcasting contents can improve the cache hit rate when the cache capacity is unlimited. Then the cache is divided into several parts through the graph partition to limit the scope of contents push. And finally, the Probability Multicast Algorithm is proposed to adapt to the change of similarity between caches. Through experiments, it is verified that our proposed methods can effectively solve the first miss problem and improve the cache hit rate. The contributions of this paper are summarized below:

First, as far as we know, this is the first to demonstrate the impact of first miss requests on the cache hit rate at the edge of the network, and we verify the existence of this problem through real datesets. Second, we propose Domain-constraint Multicast Algorithm (DMA) to proactively push content and reduce the number of first miss requests. Finally, we deal with the problem caused by dynamic changes in similarity by a Probabilistic Multicast Algorithm (PMA).

The rest of the paper is organized as follows: Section II introduces related work. In Section III, we describes the system model and introduce formally the caching problem. Graph partition model and probability model are presented in Section IV. Our evaluation results show in Section V. We conclude our work in Section VI.

II. Related Work

Researches on contents placement issues can also be roughly classify into reactive caching and proactive caching according to the order of content pushing and requests arrival.

Reactive caching: If the content is pushed after the request arrives, called reactive caching. There are many related studies in this category: [9] and [10] consider the cooperation mechanism between caches and the overhead of transferring data to make content placement decisions. Y. Hao etl. in [11] consider the computing resource requirements of transcoding between different content levels. S. Li etl. in [5] propose an online content popularity prediction mechanism according to features attached to requests. K. Poularakis etl. in [12] study the problem of base stations handover caused by user mobility in the edge cache scenario. [8] aggregates edge caches' requests for analysis and solve the small sample problem in caches.

Proactive caching, If the content is pushed before the request arrives, it is called proactive caching. The neural network is used to predict the popularity of contents and the availability of p2p nodes, and push popular contents to cache nodes in advance to reduce the traffic of the original server at the peak time [4]. [13] assumes the popularity of the contents is known, the time is divided into multiple cycles, and contents in caches are updated at the beginning of each cycle. The author gives a joint decision of caching and multicasting to minimize the expected energy cost. S.Shukla etl. in [14] consider the cost of content caching duration. With the popularity of contents is given, they propose a strategy of contents placement and the duration of contents in caches to optimize the cost caused by proactive caching and misses.

In this work, we also use multicast, but unlike the previous works, we don't know the popularity of contents, and the multicast decision is not static. Besides, the above researches do not consider the impact of first miss requests on edge caches' hit rate when caches gets closer to the users.

TABLE I
Description of frequently-used notations.

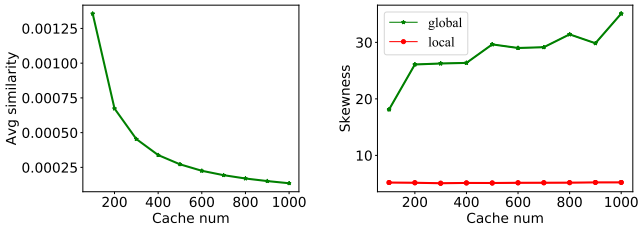
Notation	*Description*
c	content
l	edge cache
$x_{i,l,j}$	whether the global i^{th} request is the j^{th} request on l
$y_{l,j}$	whether the content of the j^{th} request on l is cached
$p_{i,l,j}$	whether to push the content of the global i^{th} request to l, which is also the j^{th} request on l.
Δk	maximum number of contents store on the cache
μ_c^l	probability of l receives a request of c
p_{uv}	probability of pushing content from cache u to cache v

III. System Model and Problem Formulation

For ease of reference, we list some notations in Table I

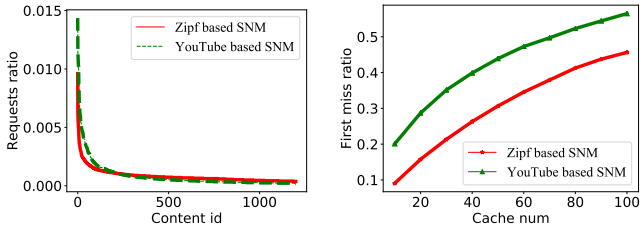
A. Motivation

Analysis of YouTube datasets [15] (which are campus requests collections) drives our work. We call each content server in the dataset as local cache, and the collection of all requests as the global view. Although content servers are not deployed at the edge of the network, they have a large number of caches just like edge caches. We sort the contents on the global / single cache in descending order of popularity, and compare the similarity between the first 10% contents sets. The similarity is based on Jaccard coefficient, $sim(C_l; C_g) = \frac{|C_l \setminus C_g|}{|C_l \cup C_g|}$, where C_l is the collection of the popular contents in cache l and C_g is the collection of the global popular contents. As shown in the Fig. 2(a), the global popular contents are not necessarily popular in local caches, so it is invalid to make decisions in local caches according to global popularity information. In Fig. 2(b), we calculate the skewness of the global content popularity distribution and the average skewness in the local caches. Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean, $Skew = E\left[\frac{X}{\quad}\right]^3$, where X is a random variable of content popularity on a cache, is it mean, is it standard deviation. We can see



(a) Similarity. (b) Skewness.

Fig. 2. YouTube Dataset analysis, the difference between request distribution in local cache and global view.



(a) Popularity distribution. (b) Proportion of first miss requests.

Fig. 3. Analysis of the requests generated by Zipf/YouTube based SNM.

that the popularity skewness on local caches is much lower, which indicates that the requests received on local caches are more arbitrary. Since requests are distributed in many caches, the percentage of first miss requests increases a lot.

Definition 1 (first miss request): If a request first appears in a cache, and there is no corresponding content in the cache, then we call the request as first miss request.

We count the distribution of requests on local caches and find that many contents on local caches were only requested once, and the number of first miss requests in the dataset is 891,121, accounting for 66.9% of the total number of requests. A large number of first miss requests seriously affect the hit rate of local caches. If there is only one cache, the number of first miss requests is 580,598, accounting for 39.7% of the total number. It means that the number of first miss requests will increase with the increase of the number of caches. Compared with the scene of the dataset, MEC needs deploy more caches at the edge of network, and the impact of first miss requests on edge caches' hit rate will be more serious. To prove this, we generated two request sequence and mapped them to different numbers of caches to simulate the requests received in caches.

We first use the SNM [6] to generate global requests sequence and then map these generated requests into each cache according to [8] to simulate the requests sequence received by edge caches in reality. In our experiment, we generated two types of requests sequences. The first is Zipf based SNM, like the practice in [8], we set the popularity of each content according to Zipf law then generate 279,880 requests for 4000 contents. The second is YouTube based SNM, we get the requests' profile from the dataset [16] instead of generating it follow Zipf law. This dataset crawls the statistical information of 64,239 videos on YouTube, including video ID, age of video, views, etc. We determine the first arrival time of each content according to the age of the video and determine the number of requests generated by this video according to the views. We map the number of requests for each content to a total of 10,000 caches according to [8], and then randomly select 100 caches, count the number of requests for each content to generate the real request sequence. We generated 403,587 requests with a total of 21,314 contents. After we generate the global requests sequence, like the practice in [8], we randomly set feature vector for content and cache, and then map requests to different caches.

We analyze the distribution of the popularity of the requests. The contents is sorted by popularity, and the id of the most popular content is 0. As shown in Fig. 3(a), the requests follows the Zipf distribution. We map these contents to different numbers of caches, as the number of caches increases, the proportion of first misses requests is also increasing. As shown in Fig. 3(b), when the number of caches increased from 10 to 100, the proportion of first miss requests to all requests has tripled. This shows that the first miss problem will become more serious as the number of caches increases.

B. System Model

Fig. 1(b) provides a system illustration. We consider a mobile network consists of base stations (BSs), each BS is endowed with a storage device that can satisfy the user's content request within its radio range. We denote the set of edge caches by $L = \{l_1; l_2; ...\}$, contains $|L|$ caches. In this paper, we do not consider the overlap between BSs, assuming that all users in a certain area will only connect to one BS. The content provider has a set of contents $C = \{c_1; c_2; ...\}$ that can be requested by the end-users, contains $|C|$ contents. Assuming that contents are the same size, and each cache storage is limited. There is an original server in the network, which contains all contents, and can decide which content to push to the cache. We denote set of requests by $R = \{req_1; req_2; ...\}$, contains $|R|$ requests, which come in sequence. Each request in this set is represented by $x_{i;l;j}$, which means that the i^{th} request in the global view is corresponding to the j^{th} request in cache l, and the content of the request is $c(i)$, $x_{i;l;j} \in \{0; 1\}$.

For each incoming request $x_{i;l;j}$, the corresponding cache l will first check the local cache. Formally, let $y_{l;j} \in \{0; 1\}$ represent whether the content $c(j)$ is cached in cache l when the j^{th} request of the edge cache arrives, $y_{l;j} = 1$ means the content $c(j)$ is cached on l. If content $c(j)$ is not found in the corresponding cache, the cache retrieves it from the original server and the original server will decide which edge caches to push this content to, we define $p_{i;l;j} \in \{0; 1\}$ indicates whether to push content $c(i)$ to edge cache l when the i^{th} request in the global arrives, j means it is the j^{th} request in cache l. When the request is missed, a content push will be triggered. We use $|L|$ dimensional binary vector P^i to represent the strategy of pushing content when the i^{th} request of global arrives, $p_{i;l;j}$ is the l^{th} dimension of the vector P^i.

(a) Unicast (b) Naive Broadcast (c) Domain-constraint Multicast (d) Probabilistic Multicast

Fig. 4. Content push strategies.

C. The Problem of Optimal Content Push Strategy

We define h_l to represent the number of hits in cache l. When the request req_i arrives, and the corresponding content is cached on edge cache l, it is calculated as a hit.

$$h_l = \sum_{i=1}^{|R|} x_{i;l;j} \quad y_{l;j} \tag{1}$$

If a cache l may cache the content requested by req_i, the content must be pushed before the request arrives.

$$y_{l;j} \quad \sum_{j_l' < j_l} p_{i;l;j_l'} \tag{2}$$

Whether the content is still in the cache or not depends on the size of the cache, the cache strategy and the order in which requests arrive. In this paper, we will not discuss the cache strategy. Assuming that the requested content was recently pushed at the $j_l'^{th}$ request, so we define the k:

$$y_{l;j} = 1 \quad \text{if} \quad j \quad j_l' < \quad k \tag{3}$$

Our objective is to find a policy $P = P^1; P^2; ::: P^k$ that maximizes the overall cache hit rate so that we can achieve the highest cache efficiency.

$$\max_P \sum_{l=1}^{|L|} h_l$$
$$\text{s.t: (1) ; (2) and (3)}$$

The timing of content push is not at the off-peak time, but after each request miss. Constraint 2 implies that a content must be cached on the server after being pushed. The k in constraint 3 mainly depends on the size of the cache, the method of content replacement, and the dynamics of the content.

IV. Similarity based Content Push Strategy

In this section, we first introduce the Naive Broadcast Algorithm (NBA), then use Domain-constraint Multicast Algorithm (DMA) to limit the scope of contents pushing, and finally improve the precision of our content push through Probabilistic Multicast Algorithm (PMA).

A. Naive Broadcast Algorithm (NBA)

To solve the first miss problem in edge caches, relevant contents must be proactive pushed to the corresponding caches before requests arriving. The simplest push strategy is broadcast, unlike unicast shown in Fig. 4(a), as long as a request misses on the cache, as shown in Fig. 4(b), the corresponding content will be pushed to all caches.

Take Fig. 1(b) as an example, there are 3 caches in the network. Contents marked in green have been requested 3 times globally and these requests evenly fall on 3 caches in a short enough time interval. If we broadcast this content to all caches in the first miss, then the subsequent 2 requests will be hit. Conversely, if we only push the content to the cache requesting this content, then all 3 requests will be missed.

Theorem 1: The variable L_c is used to represent the set of caches that have received requests for content c. If the storage capacity on each cache is infinite. Let N_f denote the max number of first miss requests that can be reduced. The expected max number of first miss requests that can be reduced is $E(N_f) = \sum_{c2C} \sum_{l2L} (1 \quad \sum_{v=0}^{V_c} (\quad _c^l)^{V_c \quad v}) \quad |C|$. The probability that a request for content c comes from cache l is $_c^l$, and $\sum_{l2L} \quad _c^l = 1$. V_c is the total number of requests for content c.

Proof 1: The max number of first miss requests that can be reduced by broadcasting is $E(N_f) = \sum_{c2C} (|L_c| \quad 1)$. It means that only the first request that arrives globally will be missed, and other caches that subsequently receive this request will not missed. The probability that the content c on BS l has been requested at least once $p_c^l = 1 \quad (1 \quad u_c^l)^{V_c}$, $E(|L_c|) = \sum_{l2L} p_c^l$. The expected maximum number of first miss requests that can be reduced is $E(\sum_{c2C} (|L_c|)) \quad |C| = \sum_{c2C} \sum_{l2L} 1 \quad 1 \quad u_c^l \quad ^{V_c} \quad |C| = \sum_{c2C} \sum_{l2L} (1 \quad \sum_{v=0}^{V_c} (\quad _c^l)^{V_c \quad v}) \quad |C|$.

NBA can solve the first miss problem in the edge cache when the cache size is unlimited. But in reality, the cache size is limited, and the broadcast strategy will bring huge overhead.

B. Domain-constraint Multicast Algorithm (DMA)

The undifferentiated broadcast will cause many caches to push some contents that will not be requested at all, which wastes network bandwidth and storage resources on the BS. Recent studies have pointed out that caching strategies can benefit from the cache location information [8] [15] [17]. We believe that the cache's preference for content is related to the

area it covers, for example, a cache covering a company may receive a request for the same content as a cache covering a coffee shop [8].

We measure the similarity between caches according to the requests received in the past. Through the different similarities between caches, as shown in Fig. 4(c), we can divide caches into different domains, when a miss occurs on cache A, the original server will push the content to caches in the same domain. We first divide the caches into two domains, construct a graph $G = (V, E)$, $V = \{l | l \in L\}$, $E = \{e_{uv} | u, v \in L\}$, each cache in L is regarded as a node in the graph. If the same request exists in two caches, there is an edge between the nodes corresponding to the two caches. The weight on each edge represents the similarity measure between node u and node v. We use Weighted Jaccard similarity (also known then as Ruzicka similarity) to measure the similarity between two nodes. Assume that $R_u = \{V_1^u, V_2^u, ...\}$ represents the collection of requests received by cache u, V_c^l means the number of requests for content c on cache l, $w_{uv} = \frac{\sum_{c \in C} \min(V_c^u, V_c^v)}{\sum_{c \in C} \max(V_c^u, V_c^v)}$.

We use Stoer-Wagner algorithm [18], with a computation complexity of $O(|V|^3)$, to solve this problem. The k-way partitioning problem can be solved by recursive bisection [19]. That is, we recursively divide each subgraph into two subgraphs. After $\log k$ rounds of recursion, we finally get k disjoint subgraphs. Thus, the problem of performing k-way partitioning is reduced to that of perform-ing a sequence of bisections. A multilevel recursive bisection (MLRB) algorithm has emerged as a highly effective method for computing a k-way partitioning of a graph. The complexity of the MLRB for a graph $G = (V, E)$ is $O(|E| \log k)$.

C. Probabilistic Multicast Algorithm (PMA)

The change of contents popularity and the movement of the crowd will lead to the change of similarity between caches. However, due to the computational overhead, the domains can-not be updated frequently. That is, at the end of each cycle, the similarity between caches may decrease signi cantly. To adapt to these changes, we propose a Probability based Multicast Algorithm to make the content push decision. We believe that in a short enough period time, the jaccard similarity between the two caches does not change much, and the de nition of jaccard similarity has natural probabilistic interpretations: Given two arbitrary caches u and v, their Jaccard's similarity is equal to the probability that a randomly chosen request of u is also a request of v [20]. Let p_{uv} denote the probability of pushing content between cache u and cache v as follows:

$$p_{uv} = \frac{\sum_{c \in C} \min(r_{uc}, r_{vc})}{\sum_{c \in C} \max(r_{uc}, r_{vc})}; \quad (4)$$

where r_{ic} represents the number of requests for content c received by cache i in a short time. We calculate the probability of pushing content for caches in the same domain, as shown in Fig. 4(d), when a request is missed on cache A, we calculate the push probability p_{AB} of cache A and cache B which in the same domain, and push content based on p_{AB}.

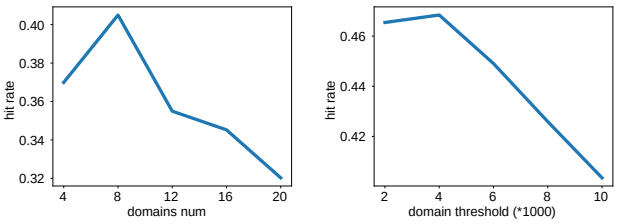
V. Experimental Results

In this section, we verify the performance of our proposed caching strategies through simulation experiments based on trajectory datasets. We compare six caching strategies: LRU, NBA, DMA, PMA, age-based threshold strategy (ABT), and ABT-prefetch strategy [8]. Due to space limitations, we did not give a detailed introduction to the request sequence generated and the parameters used in the comparison experiment. All the information can be found in our public source code [21].

The choice of the domains' number and the frequency of domain updates need to be determined according to the actual situation. Too few domains will lead to its performance close to broadcast and waste a lot of resources while too many domains will lead to its performance close to unicast. Similarly, too fast domain update frequency will lead to a lot of computing overhead, too slow will lead to a serious decline in the similarity between caches. These two values are empirical, as shown in Fig. 5, when the number of caches is 100 and the cache size is 1000, we get the optimal number of domains is 8 and the optimal frequency of domain update is 4000.

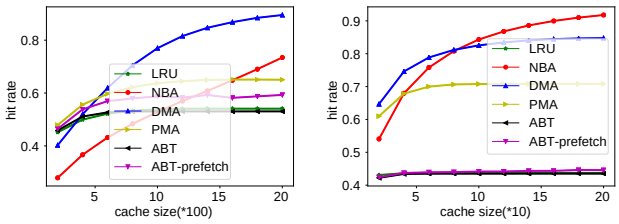
Fig. 6 shows the hit rate of different caching strategies under two requests sequences. As shown in Fig. 6(a), when the cache size is small, the performance of ABT and ABT-prefetch strategies is second only to that of PMA. This is because other push strategies frequently switch in and out of contents when the cache size is small, which leads to the drop in hit rate. With the increase of cache size, the hit rate of ABT and ABT prefetch strategies is lower than that of all the content push strategies proposed by us. In Fig. 6(b), the performance of our contents push strategies is always better than ABT and ABT-prefetch. This is because there are a large number of requests for each content in the data, resulting in a large number of concurrent requests. This simulates some special situations, such as hot news or hot games.

The proactive content push strategy also brings extra over-head. We measure this overhead by the average number of times that each request needs to transmit content. From Fig. 7, we can see that the overhead of ABT, ABT prefetch, and LRU policies is relatively low, because the frequency of content push is very low. The cost of the push strategy based on probability in Fig. 7(b) is greater than that in Fig. 7(a). This is because the request concurrency in this dataset is high, which leads to the high similarity between caches, and finally leads to the high probability of content push. In fact, due to the use of multicast technology, the real network overhead is not linear with the average number of transmissions per request. Fig. 8 shows the accuracy of contents push. When the content is pushed to a cache, if any user accesses the content before it is discarded, we think that the content push is effective. From the experimental results, we can see that compared with NBA and DMA, the PMA has the more stringent requirements on push content, which makes it more accurate than other strategies. The ABT-prefetch cache strategy needs to detect very popular content before it can start content push, so its push accuracy is the highest one.



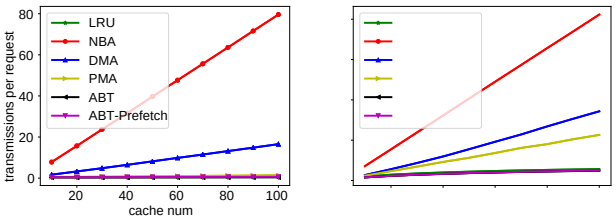
(a) Influence of domains num. (b) Influence of domain threshold.

Fig. 5. Influence of parameter selection.



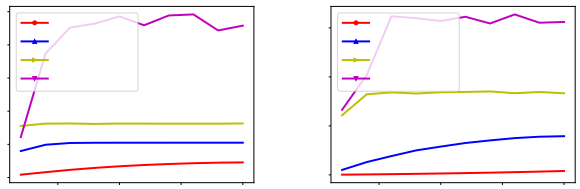
(a) Zipf based SNM. (b) YouTube based SNM.

Fig. 6. Hit rate.



(a) Zipf based SNM. (b) YouTube based SNM.

Fig. 7. Number of transmissions.



(a) Zipf based SNM. (b) YouTube based SNM.

Fig. 8. Precision.

VI. Conclusions

Based on the analysis of the dataset, this paper proposes for the first time that the first miss requests under the edge cache becomes the key factor affecting the overall hit rate of the edge cache. According to the similarity between caches rather than the popularity of content, we propose NBA, DMA and PMA. In the experiment, we use the real dataset and SNM to simulate the requests received by the edge cache, and verify that our proposed content push strategy can effectively solve the first miss problem in the edge cache under acceptable overhead, and improve the hit rate in the edge caches.

VII. Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. 62020106013, 61572113; the Science and Technology Achievements Transformation Demonstration Project of Sichuan Province of China No. 2018CC0094; and the Fundamental Research Funds for the Central Universities No. ZYGX2019J075, 2082604401036.

References

[1] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2525–2553, 2019.

[2] J. Wu, C. Yang, and B. Chen, "Proactive caching and bandwidth allocation in heterogenous networks by learning from historical numbers of requests," *IEEE Transactions on Communications*, 2020.

[3] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled wireless heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 131–145, 2015.

[4] Y. Zhang, C. Gao, Y. Guo, K. Bian, X. Jin, Z. Yang, L. Song, J. Cheng, H. Tuo, and X. Li, "Proactive video push for optimizing bandwidth consumption in hybrid cdn-p2p vod systems," in *Proc. of IEEE INFOCOM*, 2018.

[5] S. Li, J. Xu, M. Van Der Schaar, and W. Li, "Popularity-driven content caching," in *Proc. of IEEE INFOCOM*, 2016.

[6] 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 Computer Communication Review*, vol. 43, no. 5, pp. 5–12, 2013.

[7] G. S. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis, "Learning to cache with no regrets," in *Proc. of IEEE INFOCOM*, 2019.

[8] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *Proc. of IEEE INFOCOM*, 2016.

[9] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. of IEEE INFOCOM*, 2010.

[10] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1863–1876, 2015.

[11] Y. Hao, L. Hu, Y. Qian, and M. Chen, "Profit maximization for video caching and processing in edge cloud," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 7, pp. 1632–1641, 2019.

[12] K. Poularakis and L. Tassiulas, "Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 675–687, 2016.

[13] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5g wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2995–3007, 2016.

[14] S. Shukla and A. A. Abouzeid, "Proactive retention aware caching," in *Proc. of IEEE INFOCOM*, 2017.

[15] M. Zink, "Watch global, cache local: Youtube network traces at a campus network-measurements and implications," *IEEE Multimedia Computing and Networking*, 2018.

[16] X. Cheng, C. Dale, and J. Liu, "Dataset for statistics and social network of youtube videos," *Available: http://netsg. cs. sfu. ca/youtubedata*, 2008.

[17] K. Huguenin, A.-M. Kermarrec, K. Kloudas, and F. Taïani, "Content and geographical locality in user-generated content sharing systems," in *Proc. of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, 2012.

[18] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM (JACM)*, vol. 44, no. 4, pp. 585–591, 1997.

[19] G. Karypis and V. Kumar, "Multilevelk-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[20] X. Wang, W. Lu, M. Ester, C. Wang, and C. Chen, "Social recommendation with strong and weak ties," in *Proc. of ACM CIKM*, 2016.

[21] Y. L. Chao Song, "Publicly available code," *Available: https://github.com/YanpengLuo/Towards-Problem-of-First-Miss-under-Mobile-EdgeCaching*, 2021.