International Journal of Innovative
Computing, Information and Control
Volume **15**, Number **5**, October **2019**

ICIC International ©2019 ISSN 1349-4198
pp. **1655–1672**

PARTITIONING CACHING: KEEP POPULAR AND DIVERSE CONTENT IN ICN IN-NETWORK CACHING

Qifeng Yang[1,2], Haojiang Deng[1,2] and Lingfang Wang[1,2]

[1]National Network New Media Engineering Research Center
Institute of Acoustics, Chinese Academy of Sciences
No. 21, North 4th Ring Road, Haidian District, Beijing 100190, P. R. China
{ yangqf; denghj; wanglf }@dsp.ac.cn

[2]School of Electronic, Electrical and Communication Engineering
University of Chinese Academy of Sciences
No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, P. R. China

Received January 2019; revised May 2019

Abstract. *Faced with the problem of content distribution efficiency in the host-centric architecture, ICN has emerged as a promising network architecture recently. In-network cache as a core function in ICN can greatly improve the content distribution efficiency and reduce redundant network traffic. It is a challenge for ICN to design caching strategies to maximize the utilization of in-network cache. The goal of ICN caching strategy lies in 2 points: first, replicating popular contents to the edge of network to reduce the latency of downloading popular contents; second, increasing cache diversity in the neighbourhood to improve the hit ratio of cache and thus reducing traffic to the distance and avoiding high latency. Current ICN caching strategies optimize performance on either side but cannot achieve both goals at the same time for their own shortcomings. So we propose a partitioning caching strategy focusing on the trade-off between the number of caches for popular contents and content cache diversity. In partitioning caching strategy, each node has a separate popularity based cache and a collaborative cache to capture both popular and diverse content access patterns. We also propose a method for monitoring content popularity at line speed and both on-path and neighbour nodes can make cache decisions collaboratively. The simulation results show that our caching strategy can significantly reduce content retrieval latency and improve the performance of the cache network compared with other collaborative caching strategies.*

Keywords: ICN, In-network cache, Caching strategy, Partitioning caching, Popularity monitoring

1. **Introduction.** Information-centric networking (ICN) [1,2] is proposed as an effective future Internet architecture to address the problem of content dissemination efficiency in current network architecture. In-network cache plays a crucial role in ICN. Routers in the network are equipped with caching abilities to cache content items. If a content request is answered in one router (i.e., the request arrives at the router's cache) before reaching the source server, the content retrieval latency is reduced and the bandwidth of the upstream link is saved. Although web cache has been fully researched [3,4], the in-network cache is fundamentally different from the cache in web and cellular networks [5]. First, the traditional network architecture is not designed with network cache in mind. Its storage facilities such as CDN are mainly based on statically placed caching algorithm, which is too computationally intensive and cannot adapt to the dynamic needs of users and the rapid changes in in-network caching. Secondly, the traffic that ICN expects to carry will

DOI: 10.24507/ijicic.15.05.1655

cover all the content transmission of the Internet and the traffic that needs to be cached will far exceed the storage capacity of ICN in-network cache. Moreover, due to the limit of line speed [6], the capacity of the cache in ICN network will be smaller than the capacity of the storage in CDN. Therefore, it is necessary to design a simple and effective caching strategy to reasonably allocate the limited cache space in ICN.

The goal of ICN caching strategy lies in two points: first, replicating popular contents to the edge of network to reduce popular content retrieval latency; second, increasing cache diversity in the neighbourhood to improve hit ratio of cache and thus reducing traffic to the distance and avoiding high latency [7,8]. Current ICN caching strategies focus on only one point and achieve optimization only one side. For example, on-path caching strategy attempts to optimize the placement of contents and leaves copies on the path it travels between the source and the requester. Based on the Zipf distribution [9], a few popular contents in the network will occupy most of the traffic, so popular contents are more possible to be cached along the route, thus this method can effectively reduce the latency. The regional collaborative caching strategy focuses on increasing the number of different copies and maximizing content diversity within the neighbourhood cache space to reduce traffic to the source. Therefore, content requests are more possible to be served by the in-network cache and average latency can be reduced. The above strategies often sacrifice performance on another side. In the on-path caching strategy, fast and massive caching of popular contents can lead to excessive redundancy and reduced content diversity. Moreover, there is a large gap between the practical caching size and the large size of the catalog of all content objects [10]. It is impossible to cache all popular contents and thus a lot of popular contents which fail to be cached in network will result in heavy traffic to the source and additional latency. On the other hand, in the regional collaborative caching strategy, caching as many different contents as possible will evenly allocate limited space for all contents, resulting in number of popular content copies less than the number of unpopular ones due to their different quantities. Thus the latency of downloading these popular contents cannot be reduced efficiently, especially when cache space gets larger.

Our goal is to trade off the number of caches for popular contents and content cache diversity under cache capacity constraints. In our strategy, routers coordinate with both on-path nodes and one-hop neighbourhood nodes to make caching decisions. We allocate cache space for popular contents reasonably to find the suitable content diversity and to achieve a lower average latency. In order to clearly analyse the process of the trade off, we establish a cache system model to solve the caching revenue maximization problem in collaborative caching for ICN, and propose our caching strategy. To evaluate the performance of our strategy, we compare it with the state of art of on-path caching strategies and regional collaborative strategies. Real world topologies are used and various scenes are considered. The main contributions of this paper are as follows.

1) We propose a partitioning collaborative caching strategy for ICN trading off between the number of caches for popular contents and the content cache diversity to reduce content retrieval latency.

2) We formulate the caching revenue maximization problem as the average latency minimization problem in the cache system model we establish and compare the caching revenue of different caching placement strategies.

3) A new method to monitor the content popularity is designed for line speed processing requirement and dynamic popularity in ICN router. Local popularity is maintained on-line with low memory usage and operation time of $O(1)$. Overhead and feasibility are examined in our experiment.

The structure of this paper is organized as follows. Section 2 introduces related works of the state of the art in ICN caching strategy. In Section 3, we establish a cache system model and formulate the caching revenue maximization problem based on content retrieval latency, and then propose our partitioning collaborative caching strategy. The strategy and specific algorithm are described in detail in Section 4. Section 5 evaluates the performance of caching strategies and analyses the experiment results. We finally draw conclusions and point out the direction of future works in Section 6.

2. **Related Works.** The in-network caching strategies in ICN are mainly classified into three types: independent caching, on-path caching, and regional collaborative caching [11,12]. In the independent caching [13], each node independently judges the caching and forwarding of contents, which is simple and convenient but lacks information of other nodes definitely causing large redundancy and poor performance. The recent researches on caching strategies in ICN mainly focus on the on-path caching and regional collaborative caching [12,14].

Typical on-path caching such as LCD, MCD [15] can quickly replicate contents to the edge of the network. However, the problem is that as simple it is, on-path caching can generate a lot of redundancy along the route and has a high rate of replacement. WAVE [16] reforming the traditional on-path aching, can limit the spread of the less popular content caches, and the popular contents with more requests can be transmitted to the edge in batches. However, more cache redundancy begins to flood when the request frequency gets higher. Probcache [17] is to dynamically adjust the cache probability based on the distance between the caching node and the content source, and the content will be cached at the edge of the network rather than upstream with a higher probability. The above strategies do not consider the content popularity explicitly in the cache decision. Contents with high popularity are possible to be quickly replaced by contents with low popularity at any time, potentially reducing the hit ratio of cache. Bernardini et al. [18] judged whether a content is popular by maintaining a content popularity table. A content is considered popular only when the number of requests for it exceeds a given threshold. However, it simply regards the content popularity as prior knowledge without considering how to monitor it. Suksomboon et al. [19] proposed a strategy based on content popularity and distance from the requester. The most popular contents under the policy should be cached on the routers closest to the user, and the unpopular contents should be cached in other routers. However, this strategy only makes caching decisions to eliminate caching redundancy along the route. The collaborative range is limited within the route, lacking consideration of surrounding nodes even just one-hop away. Under the cache capacity constraints, such ubiquitous caching redundancy between different routes will make it difficult to effectively utilize cache resources or reduce network traffic.

In regional collaborative caching, the cache nodes comprehensively consider the state of other nodes in a wider region than the route when making cache decisions. Lorenzo et al. proposed a hash routing based strategy in [20]. When the routers in the domain receive a content request, they hash content name and determine which router is responsible for it and where to redirect it. The main idea of this strategy is that if the content exists in the cache within a domain, it will be in the specific router computed by the hash function. Heeyoung and Sunme [21] followed the hashing routing based strategy and carried out it in a similar way. Mick et al. [11] proposed multi-hop neighbourhood collaborative strategy to better utilize caches in several hops neighbourhood. In MuNCC, each cache node records the cache summary of neighbour nodes in the form of multi-level Bloom filters. Through these cache summaries, routers can determine whether the surrounding nodes have the required content cache and redirect the content request to them. In addition, when the

content is responded, if the copy of content is found through the cache summaries, it will not be stored repeatedly. So the cache redundancy can be reduced. These strategies usually include the operations of "collaborative reading" and "collaborative caching". Collaborative reading means that the node receiving the request can search the nodes out of the request forwarding path for the contents copies and redirects the request to corresponding nodes, which can decrease content retrieval latency. Collaborative caching is based on the result of collaborative reading. It means the router refuses to store the content stored by surrounding nodes repeatedly, which can reduce cache redundancy. Therefore, the regional collaborative caching usually keeps only one copy for the same content within the collaborative range to increase cache resource utilization when cache space is relatively small because small space can hold more non-repeating popular contents compared to on-path strategy so as to reduce average latency. However, these strategies are often accompanied by large computational and communication overhead. In addition, when cache space becomes a bit larger, the popular contents cannot be copied further more due to "collaborative caching" and cache space for popular contents will be limited by unpopular ones. As a result, latency cannot be reduced efficiently. In this case, [22,23] consider the reasonable placement of popular contents based on node centrality to alleviate this problem. In the node centrality based strategy, popular contents are placed in the node with higher network centrality, which usually means that the contents will be cached in a central node far away from the edge users, resulting in higher average content retrieval latency.

We found that the on-path caching replicates the most popular contents to the edge but causes a lot of redundancy in the limited cache space, resulting in additional latency for the contents fail to be cached. Regional collaborative caching reduces redundancy but it does not allocate sufficient cache space for the most popular contents, resulting in higher average latency. In this paper, we consider the main features of both strategies. We establish a model to evaluate the performance of different caching placement in the cache system, so we can conclude a reasonable caching strategy to solve the problems mentioned above.

3. **System Model.** In this section we establish a network level model of the cache system and estimate the impact of cache placement on cache system performance through the average content retrieval latency. We consider a cache network system model as shown in Figure 1.

The network consists of edge nodes $\mathbf{E} = \{E_1, E_2, \ldots, E_n\}$, an aggregation node M and a content source S. User requests can directly access the edge nodes. The aggregation node M is linked to all edge nodes and content source S forming a hierarchical topology. Here we set the cache size of each node to be the same: $C_{E_1} = C_{E_2} = \cdots = C_{E_n} = C_M$. According to the characteristics of the nodes in the ICN network, any cache node with the content copy can respond to the received content request. In our collaborative cache system, we adopt a one-hop neighbourhood collaboration method to minimize the overhead. The interaction protocol designed in routers makes nodes know existence of contents in on-path nodes and one-hop neighbour nodes. For content requests arriving at node M, if there is a corresponding copy of the content in the surrounding edge node, M can forward the request to the corresponding node, which is called collaborative reading described in Section 2. The content files available for downloading are in set $\mathbf{F} = \{f_1, f_2, \ldots, f_F\}$. The popularity of contents follows a Zipf-like distribution [9] and $\mathbf{F}$ is sorted by popularity in ascending order. The probability of requesting the kth popular content f_k is as follows, where Zipf exponent α describes the degree of deviation in file popularity:

$$P_k = \frac{k^{-\alpha}}{\sum_{i=1}^{F} i^{-\alpha}}$$

When E_i receives a request for content f_k, it first looks up the content in E_i's cache. If the cache hits, it directly responds to the request. Otherwise, the request will be forwarded along M to S as a default route. When the request passes through M, M will look up the contents in the local cache. The content f_k will be returned if a cache hit happens, otherwise M checks the cache summary of the surrounding edge node except E_i. If checking result is true, M will forward the request to the corresponding edge node, otherwise the request is sent back to the content source S.

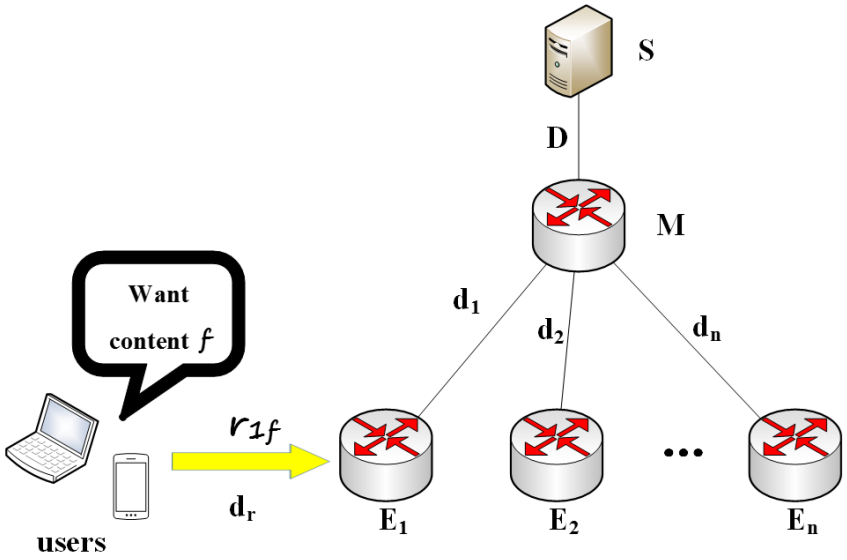

FIGURE 1. Cache network system model where **E** indicate edge nodes, M indicates aggregation node and S is content source. r_{1f} denotes the arrival rate of requests for content f from E_1.

In order to evaluate the performance of the cache placement in the cache system, we introduce the cache revenue model by calculating the average latency. Since latency and bandwidth consumption are often proportional and interchangeable, here we will focus on the revenue model for latency. The latency for the edge node to receive the content request is d_r and the latency between E_i and M is d_i. We assume that all the link latencies of the edge node E_i to M are equal: $d_i = d$. The latency of M back to the content source is D. Here we consider the latency and physical distance of the backbone network link to be larger than edge link, so we have $D = 2d_i$. Regardless of where the content is served, the latency d_r that E_i receives the content request is inevitable so we can set $d_r = 0$. Assuming that the request can always be satisfied at the content source, then the cost of retrieving contents from the source denoted by latency is $d + D$. When the request from E_i is hit in its local cache, the latency that can be saved is $d + D$. When the request is served by the cache of M, the saved latency is D. When the request passes through the M and is redirected to the neighbour edge cache E_k, the saved latency is $D - d$. The average arrival rate of content requests for item f is denoted by r_f and let r_{if} be the arrival rate of requests for content f from E_i. We assume that the request is evenly distributed among the edge nodes, i.e., $r_{1f} = r_{2f} = \cdots = r_{if} = r_f/n$. Thus, we can get a caching revenue

model denoted by saved latency:

$$C = \sum_{i=1}^{n}\sum_{f=1}^{F}(r_f/n)\left[X_{iif} * (D+d) + X_{i0f}D + \sum_{k\in B, k\neq i}^{n} X_{ikf}(D-d)\right] \tag{1}$$

$$\text{s.t.} \quad X_{ikf},\ Y_{kf} \in \{0,1\},\ \forall k \in \{0,1,2,\ldots,n\},\ \forall f \in \{0,1,2,\ldots,F\} \tag{2}$$

$$\sum_{f=1}^{F} Y_{if} \leq C_i,\ \forall i \in \{0,1,2,\ldots,n\} \tag{3}$$

$$\sum_{k=0}^{n} X_{ikf} \leq 1,\ \forall i \in \{0,1,2,\ldots,n\} \tag{4}$$

$$X_{ikf} \leq Y_{kf},\ \forall k \in \{0,1,2,\ldots,n\},\ \forall f \in \{0,1,2,\ldots,F\} \tag{5}$$

X_{ikf} is a boolean variable that equals 1 if request for content f from E_i is served by E_k and 0 otherwise. In addition, we have E_0 which denotes M for consistency. $Y = \{Y_1, Y_2, \ldots, Y_F\}$ indicates the cache distribution status of all current contents while $Y_f = \{Y_{0f}, Y_{1f}, \ldots, Y_{nf}\}$ indicates the cache distribution status of specific file f in each node. $Y_{kf} = 1$ indicates that there is a copy of f in E_k. (3) describes that the cached content in any node cannot exceed the local cache capacity. (4) indicates that there is at most one node responding to one request and (5) implies that only the node with corresponding content copy may be hit and respond. To maximize the sum of the revenues of all caches, we need to maximize the target cache revenue function (2) and find the maximum value of CR. Theorem 1 in [24] proves that this is an NP-hard problem. The global optimal solution will be accompanied by exponential computational complexity as the network size grows. It is impractical to find the most optimal solution and the solution can only be used to find static cache placement schemes under offline conditions which is not suitable for dynamic traffics in ICN. To improve time efficiency and quickly respond to traffic changes in the network, we need a lightweight, fast instant caching scheme. We consider the caching scheme from the two dimensions of content popularity and replica diversity which are considered in on-path caching and regional collaborative caching respectively as stated earlier.

1) The popular-based schemes adopt the onpath-like method and are divided into edge-popular scheme and center-popular scheme. The edge-popular scheme caches the most popular contents to the edge nodes to obtain a lower hit delay, and secondary popular contents are cached to the M. While the center-popular scheme is reversed, the most popular contents are cached in the M to get a higher hit ratio, and the secondary popular contents are cached to the edge.

2) The schemes based on cache diversity consider whether neighbour collaborative caching is required. All nodes in the collaborative scheme perform both "collaborative reading" and "collaborative caching" operation (see Section 2). If a node has already found a copy of content f in surrounding nodes (one-hop neighbourhood), it will not cache f repeatedly and thus improving the content diversity; while non-collaborative scheme performs collaborative reading only without collaborative caching operations so it can increase the number of copies for popular contents.

According to the above two dimensions, we have four cases of cache schemes: case 1: edge-popular collaborative scheme, case 2: edge-popular non-collaborative scheme, case 3: center-popular collaborative scheme and case 4: center-popular non-collaborative scheme. Then we consider the steady state content distribution of these cases as shown in Table 1. In case 1, the duplicate content will not be cached in neighbourhood of M, so the contents

TABLE 1. Steady state content distribution of 4 cases

Schemes	Edge cache	M cache
case 1	$1 \sim nC$	$nC+1 \sim (n+1)C$
case 2	$1 \sim C$	$C+1 \sim 2C$
case 3	$C+1 \sim (n+1)C$	$1 \sim C$
case 4	$C+1 \sim 2C$	$1 \sim C$

which popularity ranks from 1 to nC will be evenly cached in the nodes $E_1 \sim E_n$. The contents ranking from $nC+1$ to $(n+1)C$ are cached in M; in case 2, because there is no collaborative caching, each edge node will cache the contents ranking from 1 to C repeatedly, and the $(C+1)$th to $2C$th popular contents are cached in M due to the on-path collaboration; in case 3, the contents ranking from 1 to C are cached in M, and the contents ranking from $C+1$ to $(n+1)C$ are cached in the edge nodes; in case 4, node M cache top to Cth popular contents and each edge node will cache contents of popularity ranking from $C+1$ to $2C$. Combined with Formula (2), we get the cache revenue of the four cases as below:

$$C(case1) = \sum_{f=1}^{nC} r_f \left[\frac{1}{n}(D+d) + \frac{n-1}{n}(D-d)\right] + \sum_{f=nC+1}^{(n+1)C} r_f D \tag{6}$$

$$C(case2) = \sum_{f=1}^{nC} r_f (D+d) + \sum_{f=C+1}^{2C} r_f D \tag{7}$$

$$C(case3) = \sum_{f=C+1}^{(n+1)C} r_f \left[\frac{1}{n}(D+d) + \frac{n-1}{n}(D-d)\right] + \sum_{f=1}^{C} r_f D \tag{8}$$

$$C(case4) = \sum_{f=C+1}^{2C} r_f (D+d) + \sum_{f=1}^{C} r_f D \tag{9}$$

It is easy to find that $C(case3) > C(case1)$ and $C(case2) > C(case4)$ because r_f follows the Zipf distribution. That is to say the center-popular collaborative scheme is always better than the edge popular collaborative scheme, the edge-popular non-collaborative scheme is always better than the center-popular non-collaborative scheme, so we only need to compare case 2 and case 3 to get the best performance. We do the difference between $C(case2)$ and $C(case3)$:

$$F(C) = C(case2) - C(case3) \tag{10}$$

$$= \sum_{f=1}^{C} r_f d + \sum_{f=C+1}^{2C} r_f \frac{n-2}{n} d - \sum_{f=2C+1}^{(n+1)C} r_f \left[D - \frac{n-2}{n} d\right] \tag{11}$$

$F(C)$ monotonically increases when $C \in N+$, where n as the number of edge nodes is a constant. Here we consider the case of $n = 3$ then we have $F(1) < 0$. So there is a critical value C' to make that when $C < C'$, $F(C) < 0$ and when $C \geq C'$, $F(C) \geq 0$. That is to say, there is a critical value C' to make the center-popular collaborative scheme have a higher revenue when the cache capacity $C < C'$, and the edge-popular non-collaborative scheme performs better otherwise. To explain the result above, it is because the number of content cache in an area is limited. If all the nodes cache the most popular contents like the way edge-popular non-collaborative scheme does, the number of different content cache within the area can be rather small, which means most of the contents need to be served

by the source. If all the nodes cache content collaboratively like the way collaborative scheme does, the number of different content cache within the area can be as large as possible, but some popular contents will have to be served by neighbours instead of local cache, which also results in additional latency. So when the cache capacity C is small, the edge-popular non-collaborative scheme cannot fully utilize the limited cache space in the area, and only stores redundant popular contents in the area. While collaborative scheme can cache more different popular contents and reduces the probability of sending back requests to the content source. Thus it has a lower average latency; when the cache capacity C gets larger, both schemes can store the more popular contents in the area. However, center-popular collaborative scheme allocates more cache space for unpopular or so-called "long tail" contents. In most situations, users request collaborative cache to get hottest contents instead of getting direct responses from the nearest edge nodes, resulting in more intra-domain transmission overhead and increased latency.

So no matter we sacrifice the cache for less popular contents or most popular ones, they are neither reasonable. Therefore, we partition the storage of a node into two parts: hotspot cache specially left for the most popular contents and collaborative cache for contents different from the neighbours. Both parts hold their position to maintain a reasonable space allocation.

4. **Partitioning Caching.** Most of the previous algorithms decide content placement and partition the cache from the perspective of the network layer, such as placing popular content on the edge, or placing important content in central nodes of network. To further improve the efficiency of cache resources, we consider to allocate cache space more reasonably and propose a new partition method for a single cache node together with an efficient collaborative caching algorithm and a new popularity monitoring method to support our work. Space of each cache node is partitioned into hotspot cache for most popular contents and collaborative cache for contents different from the neighbours to solve the problem of cache space allocation between popular contents and others. In this section we will give a comprehensive description of our proposed partitioning collaborative caching strategy (PCC) and related algorithms. We first introduce the collaborative caching algorithm and the popularity monitoring method, then describe the algorithm in detail.

Collaborative Caching algorithm: Collaborative reading and caching among caches depend on the interaction of cache state. However, interaction of high frequency and large-scale cache state information will impose an extra burden on the intra-network link. So our collaborative scope is limited within on-path nodes and one-hop neighbour nodes. On-path collaborative can be realized by one-bit flag in packet like the way in LCD. And we use the Bloom filter to record the local cache content summary in each cache node and send it to the surrounding neighbour nodes periodically to perform a one-hop neighbour collaborative caching. According to the cache information, route redirection (collaborative reading) of locally unsatisfied requests can also be performed. The bits that each of the Bloom filters need to occupy are: $n = C\frac{|\ln p_f|}{(\ln a)^2}$ [25], where C is the cache capacity.

Popularity monitoring: We use the counts of local content request to represent the content popularity. A sliding window is designed based on two hash tables and a Bloom filter to record the number of content requests. In this method, the sliding window is divided into several slots (as shown in Figure 2) to dynamically maintain the latest popularity. The sliding window consists of K slots, each slot can accommodate up to N requests and keeps a Bloom filter, a threshold hash table (HT1) and a popular content hash table (HT2). The threshold hash table holds an H-bit counter recording the number of requests for all contents. Each time a content request arrives, the Bloom filter is

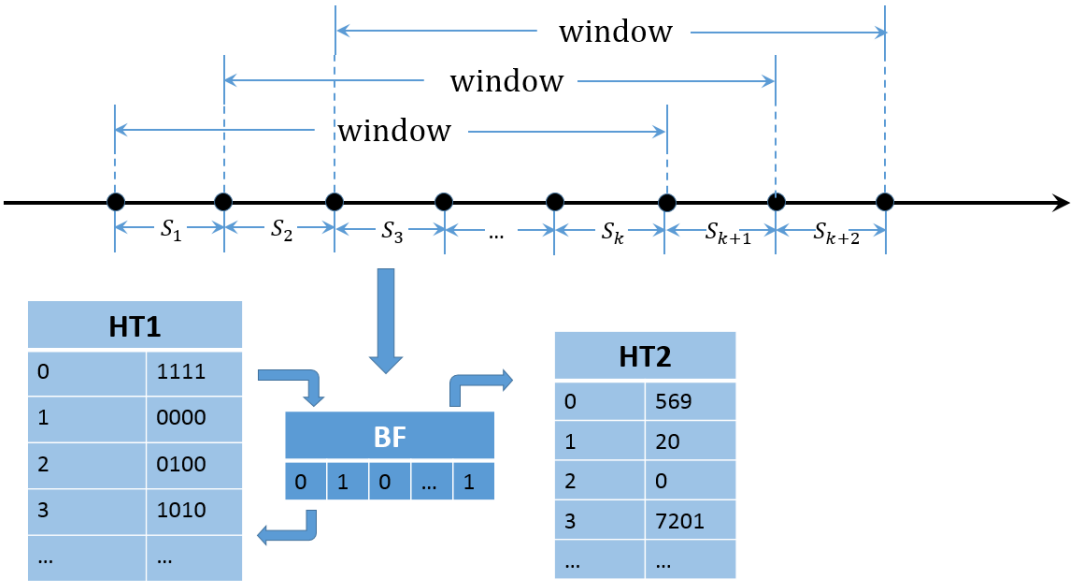

FIGURE 2. Slot-based sliding window monitoring content popularity. HT1 is a threshold hash table and HT2 is a popular content hash table. BF shunts contents to HT1 and HT2. Overflowing in HT1 will insert content into BF.

checked first. If the content has not been inserted to the Bloom filter before, it is judged as unpopular and it will be inserted into HT1. The insertion will cause the counter of the corresponding hash value added by one. Due to the design of a small H, the counter is easily to overflow, and the overflowed content will be inserted into the Bloom filter with the corresponding counter in HT1 set to be zero. When the content request arrives next time, it will be judged as a popular content worth caching. The counter of its hash value is added by one in HT2. When one slot is filled, a new slot is created and the oldest slot is deleted. The sum of the number of content counters in the last k slots is the content popularity. In addition, since the cache replacement decision is frequent, it is necessary to frequently retrieve and replace the content with the lowest popularity in the cache. The popularity of traversing the cached contents during each replacement decision will incur huge overhead, so we adopt the method in [26] by maintaining a variable $Minpop_c$ to represent the lowest popularity in local cache (both in hotspot cache and collaborative cache) to reduce overhead.

In order to increase the cache diversity and ensure that users can quickly access the most popular contents, we propose a partitioning collaborative caching strategy based on the collaboration and popularity monitoring methods above. Each node cache space is partitioned into two zones, which are a collaborative cache and a hotspot cache. The collaborative cache (CC) only caches contents that are not found in the neighbourhood or on-path nodes. The collaboration_tag field is added to the ICN packet (interest and content packet) to mark whether the content is retrieved from one-hop neighbourhood or cached in collaborative cache of other nodes. If the collaboration_tag in the content packet is one, it indicates that there is already a copy of the content in the domain, so there is no need to cache. Otherwise, it is compared with the least popular content ($Minpop_{cc}$) in local collaborative cache. If the popularity is higher than $Minpop_{cc}$, replacement happens and the content is cached. The hotspot cache (HPC) is cached by default when the space is not full. It is replaced according to local popularity when the cache is full. If the content

popularity is higher than the least popular content ($Minpop_{hpc}$) in local hotspot cache, replacement happens and content is cached. After caching and replacing in hotspot cache, the cached_tag field we designed in content packet is set to one to notice the following nodes along the route not to cache the content repeatedly. Our caching algorithm is shown in Algorithm 1 and Algorithm 2 in detail.

Algorithm 1 Interest Process Algorithm

```
Node i get a request for content
Popcount(content) ++
if content ∈ local cache then
    Response → content
else if Check_Cache_Summary(content) then
    Send request to Check_Cache_Summary(content)
    collaboration_tag → 1
end if
Forward → request
```

Algorithm 2 Content Process Algorithm

```
Cache is full and content is retrieved
if cached_tag == 0 and Minpop_hpc ≤ Popcount(content) then
    Remove → Item(Minpop_hpc)
    Cache → content
    cached_tag → 1
else if collaboration_tag == 0 then
    if Minpop_cc ≤ Popcount(content) then
        Remove → Item(Minpop_cc)
        Cache → content
        collaboration_tag → 1
    end if
end if
Forward → content
```

An example of caching decision using our algorithm is shown in Figure 3. A, B, M are caching nodes which can store at most two contents and S is the content source. The caching space of each node is partitioned into two zones HPC and CC, and either HPC and CC can only cache one content. The content popularity rate is $F_1 > F_2 > F_3 > F_4 > F_5$ in this example and the initial caching state is shown in Figure 3. First, the user 1 sends a request with collaboration tag set to 0 and cached tag set to 0 for content F_1 into the network. Cache of F_1 is hit at M and served with collaboration tag set to 1 according to Algorithm 1. When F_1 arrives at node A, according to Algorithm 2, as the popularity of F_1 is higher than F_2 in HPC and cached tag set is 0, F_1 is cached and F_2 is evicted. Finally, user 1 gets the content F_1. Then user 2 sends a request with collaboration tag set to 0 and cached tag set to 0 for content F_4 into the network. When the request is forwarded to M, M checks the caching summary of its neighbours A and B according to Algorithm 1. Then the result of checking redirects the request to node A with collaboration tag set to 1, and node A responds the request with content F_4. When F_4 is finally forwarded to B, due to the fact that the popularity of F_4 is less than F_2, HPC of node B refuses to cache it. Even though the popularity of F_4 is higher than F_5 in CC, due to the fact that

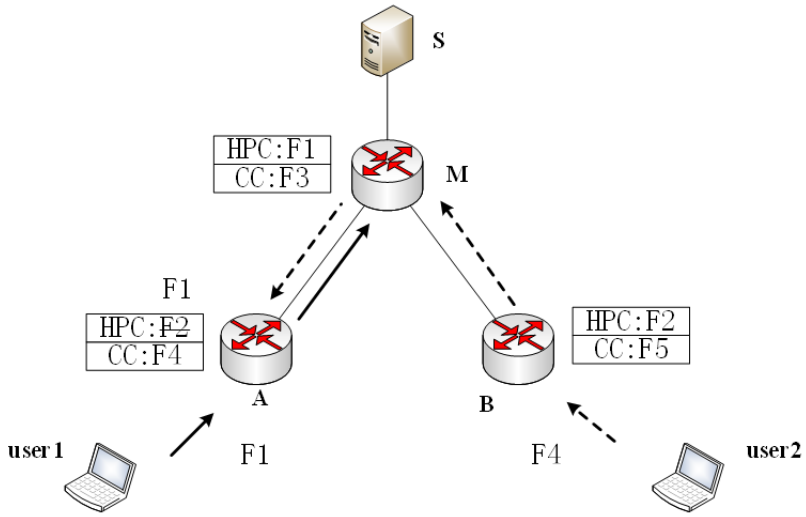

FIGURE 3. An example of caching algorithm

the collaboration tag is 1, CC of node B also refuses to cache it to avoid redundancy. At last, the F_4 is forwarded to user 2 without any caching happening.

5. Peformance Evaluation.

5.1. **Experiment setup.** We implemented PCC strategy based on Icarus [27], a simulator offering flow-level simulations for ICN caching strategy. We modified the strategy processing pipeline in the simulator and designed our own data structure for collaborative caching and popularity monitoring in the algorithm. Detailed performance analysis and overhead statistics are given below. The simulation environment is shown in Table 2.

TABLE 2. Simulation environment

Item	Key	Value
Content	size	1×10^6
	distribution	Zip-f
Cache	size	Uniformed
	replacement	LRU
Request	warmup	4×10^5
	number	1×10^6
	distribution	Poisson
	rate	100req/s

In the simulation, the stationary (IRM) workload is tested with a catalog of $N = 1 \times 10^6$ content objects. This size can be considered fair and enough to simulate a realistic scene, which has been analysed in [28]. We assume that the requests of contents are generated as $\lambda = 100$ per second following a Poisson process among all requesters. The cache of each node is empty at the beginning, and cache size of each node is equal. We spare the first 4×10^5 requests to allow caches to converge and they are not used for analysing or gathering statistics. The next 1×10^6 are logged to get statistics.

Simulation scenarios are altered in terms of network topology, Zipf exponent α, cache to population ratio and partition ratio. Zipf exponent α indicates the skewness of popularity

distribution. Cache to population ratio shows the proportion of total cache size to total content size. We will also execute the experiments with different partition ratio according to the cache size ratio and analyse our proposed algorithm.

We evaluate the performance of the strategy based on the metrics: end-to-end latency and cache hit ratio. The end-to-end latency measures the time interval from the time content request sent until the time content responded. It is one of the most important metrics for network performance. The cache hit rate refers to the portion of content requests served by in-network cache. It reflects the cache diversity and the load status of the original content server.

5.2. **Performance of PCC.**

5.2.1. *Principle verification.* We compare PCC with the edge-popular non-collaborative and center-popular collaborative caching strategy in our cache model, finding out that our strategy has a lowest average latency and a second high cache hit ratio only lower than the center-popular collaborative caching strategy. Compared to the collaborative caching strategy, PCC allocates more caching space to the popular content in hot spot cache thus reducing the cache hit ratio a bit. However, the latency of retrieving the popular contents from hotspot cache is greatly reduced thereby reducing the overall delay. Meanwhile compared to the edge-popular non-collaborative caching strategy, PCC spares more space for the popular contents which cannot be cached under edge-popular non-collaborative caching strategy, so the overall latency can be reduced and hit ratio improves. However, as the caching space increases, PCC will get a similar performance with the edge-popular non-collaborative caching strategy because both strategies have enough caching space and PCC will benefit little from additional collaborative caching space. In general, the overall hit rate is improved and the latency is reduced in PCC.

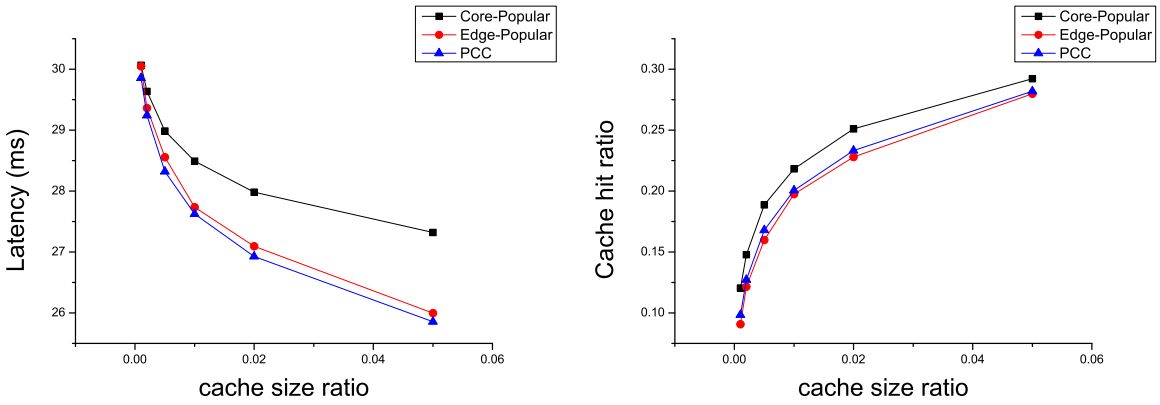

FIGURE 4. Example of figure

5.2.2. *Comparison with state of art.* We compare the PCC with LCD [15], Probcache [17], CL4M [23], HR Symm, HR Hybrid AM [20], and one-hop neighbourhood collaborative caching (CC) [11]. It can be found that in topology of our model, PCC has the lowest latency and a relatively high cache hit ratio only lower than the hash-routing based caching. This is because hash-routing based caching allows at most one cache for one content in the network which means it has the largest number of different content caching. Although the hash-routing based caching has the highest cache hit ratio, it has a large response delay due to its limited cache location and the resulting fixed routing method, which is poorer in most situations than the on-path caching and PCC. We further analyse

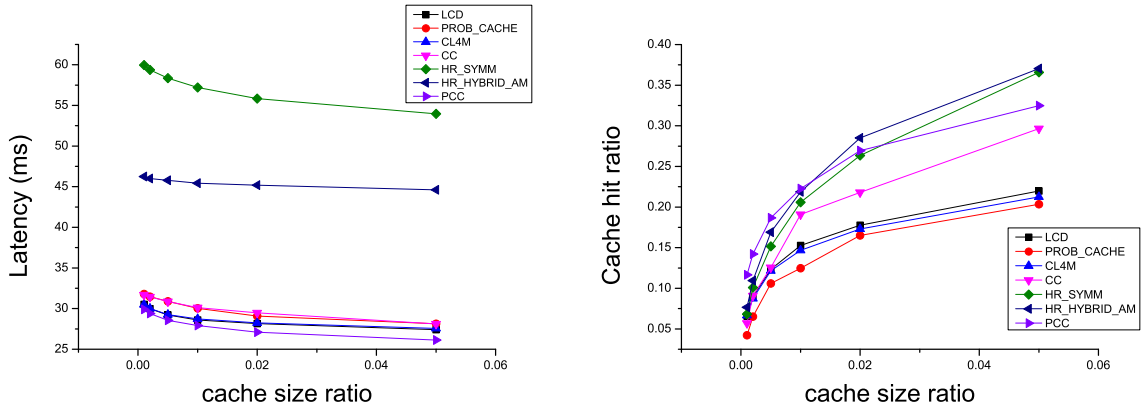

FIGURE 5. Comparison with state of art

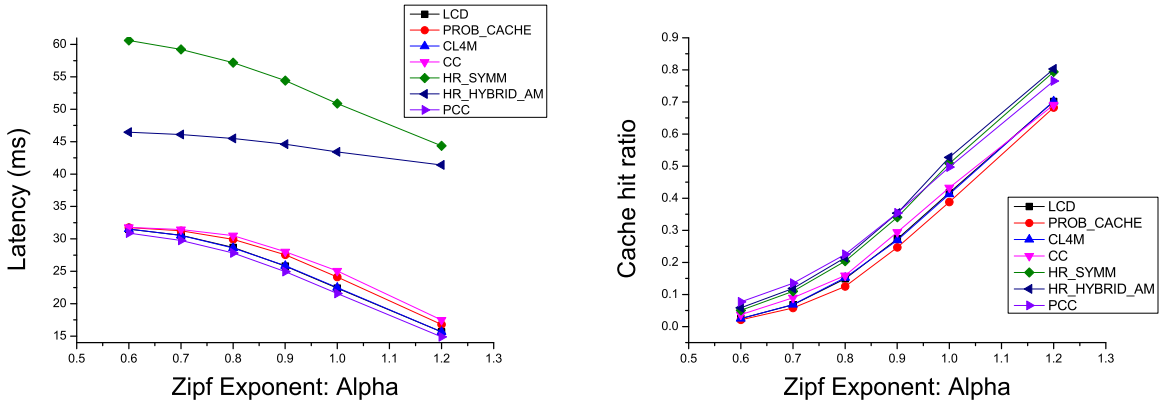

FIGURE 6. Different Zipf exponent Alpha

the performance of these caching strategies with content distribution of different Zipf exponents. As the results shown in Figure 6, regardless of the different values of the Zipf exponent, we conclude that PCC always has the lowest latency among all the strategies mentioned above. The performance attenuation of PCC is also the smallest, which means the performance is stable like other on-path caching strategies but the PCC strategy has lower latency and better performance.

5.2.3. *Performance in different topologies.* We will further analyse the performance of various strategies in different scenarios by using real-world topologies: GEANT (European academic network), GEANT2 (only part of the nodes in GEANT are selected to provide caching), WIDE (Japanese academic network) and AS1775 (EBONE, Europe). GEANT and WIDE set the node with degree 1 as the edge node, directly connecting to the users while nodes in other topologies can all generate user requests, which is much closer to the ubiquitous caching network in ideal ICN. GEANT2 considers the progressive deployment of ICN, only some nodes in the network provide ICN intra-network caching.

It can be found that in the real network topologies, the hash routing based caching strategy still has the highest cache hit ratio, meanwhile, the hit ratio and latency performance of HR Hybrid AM are both better than those of HR Symm. In most cases, the PCC has a lower hit ratio than the traditional hash caching and neighbourhood collaborative caching. However, in all cases, PCC has a lowest latency. In addition, the on-path caching strategy has a lower latency than the traditional collaborative strategy. This is mainly due to the unreasonable routing method of the traditional collaborative

strategy. Our proposed PCC uses the hotspot caching method to reduce the possibility of this unreasonable routing, as well as ensuring the cache hit ratio, achieving the lowest latency compared to other strategies. What is more, PCC outperforms other strategies in both GEANT and WIDE but there is a bit difference in AS1775. This is mainly because PCC is more suitable for hierarchical network structure. Because GEANT and WIDE have a hierarchical structure composed of edge nodes and central nodes, PCC can further reduce the average latency by leaving the content caching hierarchically along the way compared to the situation in AS1775.

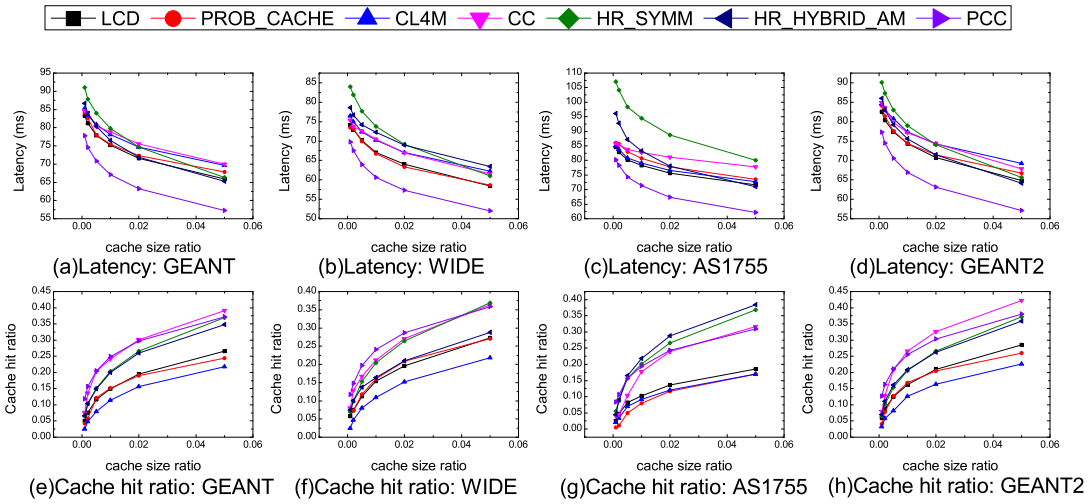

FIGURE 7. Performance in different topologies

5.2.4. *Analysis of partitioning ratio.* We will analyse the impact of different partitioning ratios (hot-pot cache/collaborative cache) on the caching performance. As shown in Figure 8, these are overall average latency of different cache partitioning ratio under given cache size ratio respectively. It can be found that as the cache partitioning ratio increases, the latency gradually decreases until reaching a minimum value, then it rises slowly. We infer that there is a best partitioning ratio to minimize the latency at the bottom of the curve. The inference demonstrates our theory in Section 3 that the content retrieval latency can be effectively reduced through reasonable cache space allocation for popular contents and others. As shown in the figure, when the cache size ratio is 0.001, the optimal partitioning ratio is between 0.1 and 0.5. When the cache capacity is 0.002 and 0.05, the optimal partitioning ratio is between 0.5 and 2. The trend can be roughly observed from the curve that the best partitioning ratio gradually increases with the increase of the cache capacity. So we conclude when the cache space increases, the impact of collaboration is reduced because the latency led by the additional routes will increase, and we need to allocate more space for popular contents as the cache capacity increases.

5.3. **Overhead and performance of data structure.** According to the collaborative caching algorithm and the popularity monitoring method, the strategy proposed in this paper has greatly improved the performance of the cache network, but it also brings additional overhead and network load. We will evaluate the overhead of the algorithm from the aspects of collaborative caching mechanism and popularity monitoring method.

5.3.1. *Collaborative overhead.* We will show that the false positive events of the Bloom filter used for cooperation have little impact on caching performance and have low memory overhead. The standard false positive probability of the Bloom filter we allowed in the

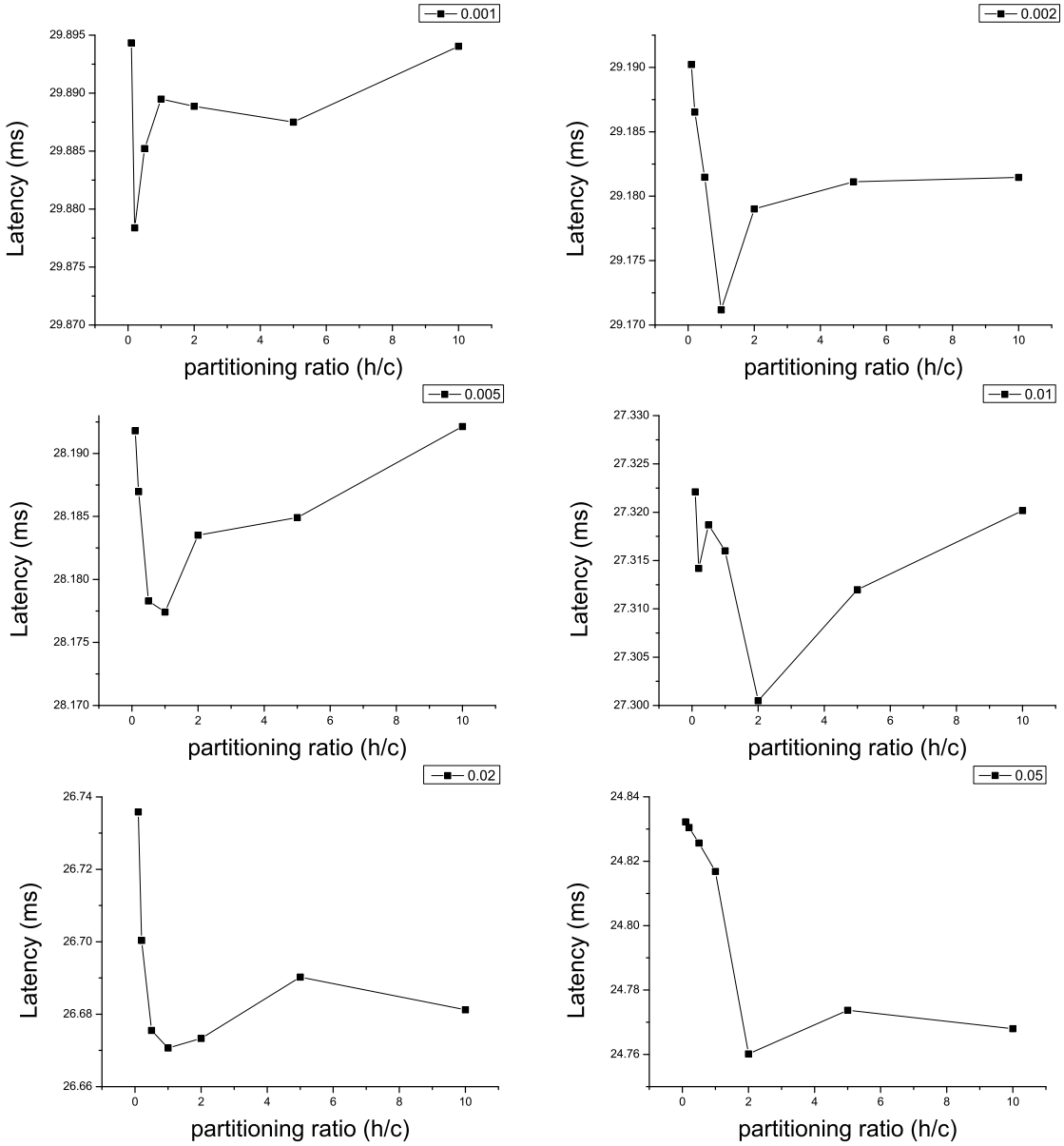

FIGURE 8. Effect of partitioning ratios

experiment is 0.02. The false positive probability in each topology refers to the ratio of the request goes to the potential cache node but missing in the overall requests. The experimental results are shown in Table 3. False positives may be caused by cache eviction or false positives of Bloom filters. In the case of exchanging BF per second, it can be found that the false positives decrease slightly when the cache space rises. Therefore, the false positive of Bloom filter plays a leading role in the ratio of overall false positive events. From Table 3, WIDE has a lowest false positive probability, GEANT is the second, while TISCALI and AS1755 have higher false positive ratios. This is because the Bloom filter of each node is independently queried. When a single node queries the BF of the surrounding nodes, the relationship between the false positive probability f and the standard false positive probability p is $f = 1 - (1 - p)^{n-1}$. So the larger the node degree is, the higher the false positive probability will be. TISCALI and AS1755 have higher node degrees and a larger network size which means longer transmission paths, so false positives are higher than that of WIDE and GEANT.

TABLE 3. Request false positive of different topologies

Topology	Cache size ratio	False positive ratio
GEANT	0.001	0.031686803
	0.01	0.033405196
	0.1	0.030051487
GEANT2	0.001	0.033263584
	0.01	0.033054122
	0.1	0.029964229
WIDE	0.001	0.016168018
	0.01	0.015602727
	0.1	0.013727368
TISCALI	0.001	0.043806633
	0.01	0.042633660
	0.1	0.034625508
AS1755	0.001	0.061019130
	0.01	0.054842105
	0.1	0.054879816

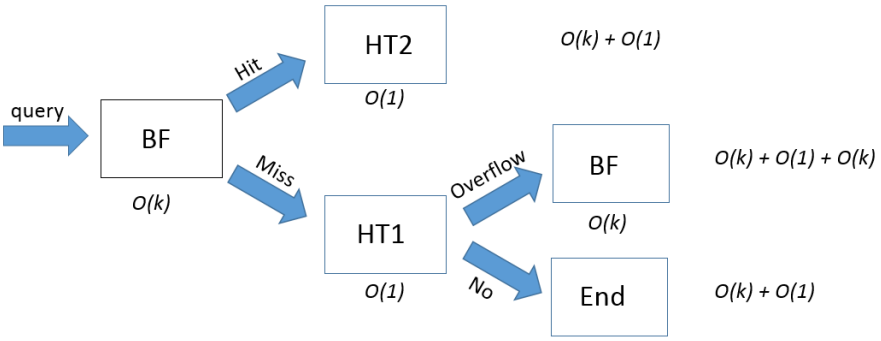

FIGURE 9. Pipeline and computational complexity of data structure

The cache capacity of each node is C (entries) and the space required for each node to hold the Bloom filter for the local content cache is $n = C * \frac{|\ln p_f|}{(\ln a)^2} \approx 8C$ bits. The ICN cahing unit is usually a 1MB sized chunk [29]. In the case of current memory size of 10GB, we have C =10GB/1MB = 10000 so each BF size is about 10kB, and the link load for exchanging BF once per second is 80kbps. The bandwidth occupied by the link of 100Mbps is negligible. The 10kB-BF can be built in the SRAM, and the processing delay is about 0.45ns [30], which can ensure the line speed processing of ICN packets.

5.3.2. *Popularity monitoring overhead.* We will analyse the computational complexity of monitoring popularity in this section. The processing pipeline is shown in Figure 9. k is the number of hash functions that needs to be calculated when querying BF. Therefore, the time complexity of the worst case in a process is $O(k)+O(1)+O(k)$, that is, querying BF, inserting a hash table then inserting BF. The process of calculating the hash can be parallel executed, so a router needs to access the BF twice and access the hash table for one time during popularity process of one content. Then we analyse the spatial complexity of this algorithm. For the slot we designed, we can monitor the popularity of latest ten thousand content requests. As shown in [31], more than 90% of contents can be filtered out by a small counter. So we need $N \approx 10000/10*3/0.5 = 6$Kbit BF to filter

the contents. The space utilization efficiency of hash Table 1 can be 50% and each item has a 4-bit counter, then each hash Table 1 in the slot will occupy 1W*4/0.5 = 80kb, the hash Table 2 counter will be log10000 = 13 bits with 50% space utilization efficiency, so hash Table 2 will occupy $(10000/10) * 13/0.5 = 26$kb. If we will build BF in SRAM, hash table in RLDRAM, according to [30], the popularity processing time of chunk will be 15.9ns. Considering the change of chunk size, in order to ensure that 10kB chunks (which is relatively small) can be processed at 100Gbps link line speed, there can be up to 50 slots at the same time. Our design deploy 10 slots. The slot scheme ensures that the popularity of chunks can be monitored at line speed and data structure only occupies a reasonable memory which can be deployed immediately.

6. **Conclusion and Future Work.** In this paper, the problem about the contradiction between the number of caches for popular contents and content cache diversity is raised. We formulate a caching revenue maximization problem to analyse this contradiction in the cache system model we establish. Then we present the partitioning collaborative caching strategy where cache space in ICN routers is partitioned into hotspot cache space and collaborative cache space to solve the contradiction. PCC is simple and convenient to deploy and has low collaborative overhead. From the results of our simulation, it can significantly reduce content retrieval latency and get a relatively high cache hit rate compared with state of the art.

In the future, we will further study how to choose a reasonable partitioning ratio adaptively and design new coupling mechanisms between the hot cache and the collaborative cache to improve the cache utilization. In addition, due to the current progress of ICN routing technology [32], we will further study the ICN-based routing mechanism based on our platform and implement our caching algorithm in a more realistic scenario.

Acknowledgement. This work was supported by "Research and Development of 5G and Information-Centric Networking (ICN) Integration Technology" National Science and Technology of Major Projects (No. 2017ZX03001019).

REFERENCES

[1] V. Jacobson et al., Networking named content, *Proc. of the 5th International Conference on Emerging Networking Experiments and Technologies*, Rome, Italy, pp.1-12, 2009.

[2] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros and G. C. Polyzos, A survey of information-centric networking research, *IEEE Communications Surveys & Tutorials*, vol.16, no.2, pp.1024-1049, 2014.

[3] J. Dai, Z. Hu, B. Li, J. Liu and B. Li, Collaborative hierarchical caching with dynamic request routing for massive content distribution, *2012 Proceedings IEEE INFOCOM Workshops*, Orlando, FL, USA, pp.2444-2452, 2012.

[4] H. Hu, Y. Li and Y. Wen, Toward rendering-latency reduction for composable web services via priority-based object caching, *IEEE Trans. Multimedia*, vol.20, no.7, pp.1864-1875, 2018.

[5] C. Lee, An optimal resource allocation scheme for D2D communications underlaying 5G-based cellular networks, *ICIC Express Letters*, vol.12, no.7, pp.731-736, 2018.

[6] A. Arianfar and P. Nikander, Packet-level caching for information-centric networking, *Proc. of the Re-Architecting the Internet Workshop (ReArch 2010)*, http://users.piuha.net/blackhawk/contug /cache.pdf, 2010.

[7] G. Zhang, Y. Li, T. Lin et al., Survey of in-network caching techniques in information-centric networks, *Journal of Software*, vol.25, no.1, pp.154-175, 2014.

[8] B. A. Ramanan et al., Cacheability analysis of HTTP traffic in an operational LTE network, *Proc. of Wireless Telecommun. Sysmp.*, Phoenix, AZ, USA, pp.1-8, 2013.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, Web caching and Zipf-like distributions: Evidence and implications, *Proc. of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol.1, no.1, pp.126-134, 1999.

[10] A. Anand, C. Muthukrishnan, A. Akella et al., Redundancy in network traffic: Findings and implications, *ACM SIGMETRICS Performance Evaluation Review*, vol.37, no.1, pp.37-48, 2009.
[11] T. Mick, R. Tourani and S. Misra, MuNCC: Multi-hop neighborhood collaborative caching in information centric networks, *Proc. of the 3rd ACM Conference on Information-Centric Networking*, Kyoto, Japan, pp.93-101, 2016.
[12] L. Huang, Y. Guan, X. Zhang and Z. Guo, On-path collaborative in-network caching for information-centric networks, *Proc. of 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, pp.175-180, 2017.
[13] H. Wu, J. Li, J. Zhi et al., Design and evaluation of probabilistic caching in information-centric networking, *IEEE Access*, vol.6, pp.32754-32768, 2018.
[14] J. A. Khan, C. Westphal and Y. Ghamri-Doudane, A content-based centrality metric for collaborative caching in information-centric fogs, *Proc. of 2017 IFIP Networking Conference (IFIP Networking) and Workshops*, Stockholm, Sweden, pp.1-6, 2017.
[15] L. Nikolaos, C. Hao and S. Ioannis, The LCD interconnection of LRU caches and its analysis, *Performance Evaluation*, vol.63, no.7, pp.609-634, 2006.
[16] K. Cho, M. Lee, K. Park et al., WAVE: Popularity-based and collaborative in-network caching for content oriented networks, *2012 Proceedings IEEE INFOCOM Workshops*, Orlando, FL, USA, pp.316-321, 2012.
[17] I. Psaras, W. K. Chai and G. Pavlou, Probabilistic in-network caching for information-centric networks, *Proc. of the 2nd ICN Workshop on Information-Centric Networking*, Helsinki, pp.55-60, 2012.
[18] C. Bernardini, T. Silverston and O. Festor, MPC: Popularity-based caching strategy for content centric networks, *Proc. of 2013 IEEE International Conference on Communications (ICC)*, Budapest, Hungary, pp.3619-3623, 2013.
[19] K. Suksomboon, S. Tarnoi, Y. Ji et al., Popcache: Cache more or less based on content popularity for information-centric networking, *Proc. of the 38th Annual IEEE Conference on Local Computer Networks*, pp.236-243, 2013.
[20] S. Lorenzo, P. Ioannis and P. George, Hash-routing schemes for information centric networking, *Proc. of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking*, Hong Kong, China, pp.27-32, 2013.
[21] J. Heeyoung and K. Sunme, A multiple hash routing scheme for fast data retrieval in ICN, *Proc. of 2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, South Korea, pp.1562-1565, 2018.
[22] J. Wang, J. Zhang and B. Bensaou, Intra-AS cooperative caching for content-centric networks, *Proc. of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking*, Hong Kong, China, pp.61-66, 2013.
[23] D. Rossi and G. Rossini, On sizing CCN content stores by exploiting topological information, *2012 Proceedings IEEE INFOCOM Workshops*, Orlando, FL, USA, pp.280-285, 2012.
[24] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos and L. Tassiulas, Caching and operator cooperation policies for layered video content delivery, *Proc. of IEEE INFOCOM 2016 – The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, USA, pp.1-9, 2016.
[25] P. S. Almeida, C. Baquero, N. Preguica and D. Hutchison, Scalable bloom filters, *Information Processing Letters*, vol.101, no.6, pp.255-261, 2007.
[26] X. Zhu, J. Wang, L. Wang and W. Qi, Popularity-based neighborhood collaborative caching for information-centric networks, *Proc. of 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, San Diego, CA, USA, pp.1-8, 2017.
[27] L. Saino, I. Psaras and G. Pavlou, Icarus: A caching simulator for information centric networking (ICN), *Proc. of ICST SIMUTOOLS*, Lisbon, Portugal, pp.66-75, 2014.
[28] V. Sourlas, I. Psaras, L. Saino and G. Pavlou, Efficient hash-routing and domain clustering techniques for information-centric networks, *Computer Networks*, vol.103, pp.67-83, 2016.
[29] L. Wang, S. Bayhan and J. Kangasharju, Optimal chunking and partial caching in information-centric networks, *Computer Communications*, vol.61, pp.48-57, 2015.
[30] D. Perino and M. Varvello, A reality check for content centric networking, *Proc. of the ACM SIGCOMM Workshop on Information-Centric Networking*, Toronto, Ontario, Canada, pp.44-49, 2011.
[31] G. Zhang, J. Zhang, B. Wang et al., On-line popularity monitoring method based on bloom filters and hash tables for differentiated traffc, *China Communications*, vol.6, no.s1, pp.72-86, 2016.
[32] G. Zhu and Y. Tian, Research progress and outlook of routing technique in information center network, *Journal of Network New Media*, vol.7, no.5, pp.1-5, 2018.