P-CLS: A Popularity-driven Caching Location and Searching Scheme in Content Centric Networking

Yuemei Xu*, Shuai Ma†, Yang Li‡, Fu Chen* and Song Ci§
* Department of Computer Science, Beijing Foreign Studies University, China
† Viterbi School of Engineering, University of Southern California, USA
‡ Institute of Information Engineering, Chinese Academy of Science, Beijing, China
§Department of Computer and Electronics Engineering, University of Nebraska-Lincoln, USA

***Abstract*—Content Centric Networking (CCN) is an emerging next-generation network infrastructure around content dissemination and retrieval, shifting from physical locations to named data. The built-in caching capacity of CCN, termed as in-network caching, promises to enable fast and effective content distribution at a global scale. Because of the in-network caching, the caching strategy of content location potentially affects how to make content searching decisions, while content searching in return decides in which routers content are stored. The relationship between content caching location and content searching has not been fully exploited in CCN or further used for the whole network performance improvement. This paper exploits the content caching location and content searching mechanism of CCN, and proposes a Popularity-driven content Caching Location and Searching scheme (P-CLS) in CCN. P-CLS leverages content access popularity to realize diverse content distribution and reduce content caching redundancy, which also overcomes the *oscillation* and *frequent replacement* phenomenon in the existing content caching and searching (CLS) scheme. Extensive simulations via hierarchical and arbitrary caching topologies show that the proposed scheme outperforms the existing caching algorithms.**

I. Introduction

Millions of multimedia data (e.g., user-generated content, Video-on-demand, HTTP web pages) are generated and shared by content producers and consumers. This trend has been posing high stress on network bandwidth and content storage, resulting in network congestion and server overload. To address these issues, Content Delivery Network (CDN) and application-specific solutions like peer-to-peer (P2P) are popular. However, CDN may experience sub-optimal performance due to the traffic engineering of Internet service provides (ISPs) [1] and P2P systems incur a lot of inter-ISP traffic and are unstable in terms of content availability and download performance [2].

To overcome limitations of CDN and P2P, accelerate network data delivery and reduce server load, Content Centric Networking (CCN) has been proposed as a predominant next-generation Internet architecture [3], which is funded upon the idea that most users only focus on the accessing data, rather than the physical locations from which the data are retrieved. `In-network caching`, as an intrinsic component of CCN, enables each router in CCN to equip a content store (CS) module (we call these routers as C-routers for short), attempting to maximize the probability of content sharing while satisfying users' requests as close to end-users as possible.

Due to `In-network caching`, two important issues need to be addressed in CCN. First, C-routers become available containers to cache content for the purpose of satisfying the subsequent requests. Thus we need to decide which C-routers in the content delivery path should cache the content, in order to maximize the utilization of CS while maintaining the content diversity of network. That is what to cache, what to replace and where to cache, and is defined as a **content caching location** problem. Second, content replicas in C-routers change with time, which is because when a C-router needs to cache content but its CS is full, it will carry out a content replacement strategy (e.g., LRU or LFU) to evict content for the new coming one. Due to the high volatility of content in CS, C-routers take efforts to find and select an appropriate content replica location to forward requests. It is known as **content searching** problem.

The content caching location and content searching problems have not been fully exploited in CCN and are usually studied separately. On one hand, two research lines exist in *content caching location* problem, namely *on-path* and *off-path* content caching. For example, Leave Copy Everywhere (LCE), Leave Copy Down (LCD) and Move Copy Down (MCD) are all the *on-path* content caching strategies, where content may be cached by any on-path cache or a subset of traverse caches in its delivery path. In contrast, *off-path* content caching often calculates the (near) optimal content placement off-line. *Off-path* strategies can achieve better network performance than *on-path* ones but at the cost of time complexity. On the other hand, *content searching* in CCN is usually implemented coordinately in a distributed manner, relying on local cache management policies as well as the relative position of caches in the network to achieve good performance. As the Internet traffic grows dramatically, it is still very challenging to forward user requests towards a "best" (e.g. closest) available replica in CCN.

*This work is supported by the Fundamental Research Funds for the Central Universities (No.023600-500110002), the National Natural Science Foundation of China (No.61502038, No.61170209) and Program for New Century Excellent Talents in University (No.NCET-13-0676).
Corresponding email: xuyuemei@bfsu.edu.cn.

978-1-4673-8590-9/15/$31.00 ©2015 IEEE

Recently, there is a tend to exploit the **content caching** location and **content searching** problems tightly, suggesting that *content searching potentially decides request forwarding path and in which C-routers content are stored, while content caching location in return affects how to make content searching decisions for forwarding requests appropriately.* Among these researches, Li *et al.* proposes an implicit coordinate content location and searching scheme (CLS) and shows promising potential in this research direction [4]. However, CLS scheme is designed in a hierarchical caching topology, but not the arbitrary caching topology of CCN, which will lead to a *oscillation* and *frequent replacement* phenomenon when applying it in CCN. The reason lies in that CLS makes the content caching down or up decisions only considering the current hitting event while ignoring the global content popularity.

In this paper, we focus on the content caching location and searching problem in an arbitrary caching topology of CCN, and novelly propose a Popularity-driven content Caching Location and Searching scheme (P-CLS). Researches [5], [6] reveal that content popularity is (by far) the most important factor affecting the network performance, compared with content request distributions, content catalog size, cache replacement strategies and etc. P-CLS distributes content towards end-users considering content popularity as well as the content caching location.

The main characteristics of P-CLS are summarized as follows:

1) **Popularity-based:** Through incorporating the content popularity factor into the content caching location and searching decisions, P-CLS not only realizes to distribute content to these C-routers which request them most frequently, but also prevents the *oscillation* and *frequent replacement* phenomenon of CLS.
2) **Content Diversity:** In P-CLS, there is always one and at most one copy of a content cached on the whole path between a server and an edge C-router. Thus, P-CLS can make more efficient cache utilization while guaranteeing content diversity in the whole network.
3) **Trail-based Searching:** P-CLS creates a trail to store the content caching history during the content cached up and down. The trail and content popularity information helps to find a "best" (e.g. closest) available replica arbitrary caching topology of CCN.

The reminder of this paper is organized as follows. Section II presents a survey of the related work and the background. Section III describes the relevant problem analysis. Then section IV presents the system model and section V describes the proposed P-CLS scheme in detail. Section VI presents the simulation results and conclusions are summarized in section VII.

II. Related Work and Background

In-network caching plays an important role in speeding up content delivery, reducing network traffic and alleviating server load, as it can cache the content passed by it and push popular content to edge network closer to end-users. Recently, a large body of researches focus on how to improve the efficiency of in-network caching in CCN, such as exploring optimal replication strategy [7], [8], explicit cooperative caching [9], [10], implicit cooperative caching [11]–[13], cache-aware routing [14] and bandwidth and storage sharing mechanism [15], [16]. All of these studies consistently reveal that content caching location and content searching mechanism are the essential parts of in-network caching, therefore need to be sophisticatedly designed and deliberated.

A. Content Caching Location

An implicit and transparent approach towards content caching location called *Leave Copy Everywhere* (LCE) [17] is used in CCN by default. LCE places content copies at all the intermediate C-routers from the hitting C-router to the requested end-user. LCE can realize fast content delivery but will lead to unnecessary content caching redundancy. LCD [2] and MCD [2] are implicit coordinated content caching strategies frequently used in Web caching systems or CDN, and have been introduced to CCN. In LCD, the requested content hit at l level will be pulled down to l-1 level. Simultaneously, the l level node keeps the requested content. MCD is similar to LCD and the only difference is that the l level node will delete the requested content after it has been moved to l-1 level.

Studies [18] show that LCD works better than MCD as it can cache one more copy on-path to serve clients from other branches. Furthermore, LCD also outperforms LCE and greatly reduces content caching redundancy [18], and thus is used for comparison in our simulations.

B. Content Searching

Along the research line of content searching, a best-effort approach called *Breadcrumbs* is proposed to use the content caching location information to improve the efficiency of content searching [19]. In *Breadcrumbs*, a trail for the purpose of storing content forwarding history is created and maintained at each router when the content is downloaded. Thus the subsequent request for the same content may be routed towards the nearest content copy directed by such a trail. Sourlas *et al.* propose an intra-domain cache-aware content searching scheme that computes the paths with minimum transportation cost based on the information item demands and the caching capacities of the network [14]. Chiocchetti *et al.* compare two CCN content searching strategies: a deterministic exploitation of forwarding requests towards a "known" copy and a random request flooding exploration towards an "unknown" copy [9]. Among these content searching strategies, *Breadcrumbs* is superior as it can take a good balance between implementation complexity and network improvement benefit.

Li *et al.* was the first one to exploit the content caching location and content searching in a tight manner and proposed a simple content caching location and searching scheme called CLS [4]. The basic idea of CLS is that the hit content at level l is pushed down to the l-1 level router towards the end-users, and the evicted chunk at level l is pushed back to the l+1 level

cache. At the same time, the trail storing the chunk history is set up during the content cached up and down, which can be used to direct the following content search. More details about CLS can be found in [4].

As we can see, CLS skillfully takes the advantages of LCD and *Breadcrumbs*, trying to reduce content caching redundancy and direct requests by historical trail information, therefore can achieve a better network performance than LCE and LCD strategies.

C. Content Popularity

Content popularity is considered as the most important factor that affects the network performance [5], [6]. Bernardini [20] *et al.* consider the high skewness of content popularity and propose a caching strategy named Most Popular Content (MPC). In MPC, every C-router counts the local number of requests for each content and stores the pair information (content name; popularity count). Although MPC outperforms LCE, LRU and LFU strategies, it has undesirable features of slow convergence of hitting rate and unstable hitting rate performance for various cache sizes. In order to overcome weak points of MPC, Mau *et al.* have proposed a Fine-Grained Popularity-based caching (FGPC) strategy [6]. Li *et al.* propose two popularity content caching algorithms, named TopDown and AsympOpt from the perspective of optimization [21]. They formulate the content caching problem and then solve it to lead C-routers coordinately make online caching decisions independently. Cho *et al.* consider the content popularity as well as inter-chunk relation and then propose a popularity-based content caching strategy called WAVE [2]. An upstream node in WAVE recommends the number of chunks to be cached at its downstream node based on the content popularity.

All these studies show promising results by incorporating content popularity into content caching decisions, but not referring to content searching. In this paper, we try to incorporate it into the tight couple problem of content caching decision and searching problem.

III. Problem Analysis

CLS shows a promising result in the direction of investigating content caching location and searching tightly. However, CLS also has its own problems. Originally, CLS is designed in a hierarchical network topology and will cause undesirable *oscillation* and *frequent replacement* phenomenons when applying it in the arbitrary caching topology of CCN. Fig. 1 shows the difference between hierarchical and arbitrary network topologies. The hierarchical caching network topology is often deployed in CDN or Web caching systems. In contrast, in-network caching in CCN makes caches ubiquitous, therefore C-routers may cover the whole Internet scale and are constructed in an arbitrary network topology. The fixed parent-child relationship presented in hierarchical topologies vanishes in CCN, which leads to the *oscillation* and *frequent replacement* phenomenon.

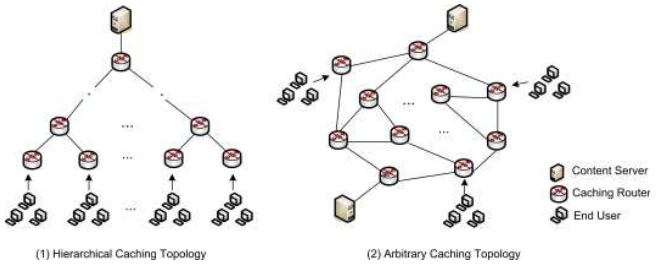


Fig. 1. Hierarchical and Arbitrary Caching Topologies.

A. Oscillation Phenomenon

Fig. 2 shows an example of oscillation phenomenon of CLS. A network consists of 2 servers and 3 routers. Server 1 is the original source for content B and G, and server 2 is for content R. For simplicity, we assume that all the 3 content have the same unified size and all routers have cache size of one content. In this case, the content caching location and the trail information are shown in Fig. 2.

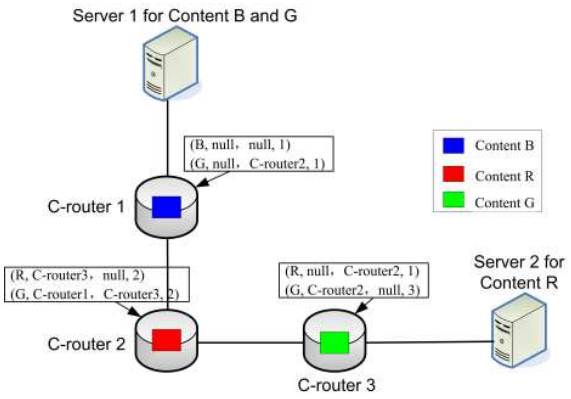


Fig. 2. An Example of Oscillation Phenomenon of CLS.

Now there is a cache hit at C-router 1, causing content B to be sent down to C-router 2. Then C-router 2 has to evict content R to make room for content B. In an hierarchical topology, the content B and R would just change places. But in this case, the evicted content R is sent up towards its content server, which is Server 2. Next, content G in C-router 3 is evicted to make room for content R, and content G is sent down to C-router 2. Content G will be cached in C-router 2, evicting the content B, which will be sent up to C-router 1 once more. As such, there has not 2 but 6 cache changes, and the state of content B, the one requested originally, is not changed after all swaps take place. As the function of the network topology, the number of such cache content changes may grow indefinitely, which is defined as the *"Oscillation Phenomenon"*.

B. Frequent Replacement Phenomenon

CLS proposes to push the evicted content at level l back to the l+1 level cache in any situation, which may lead to

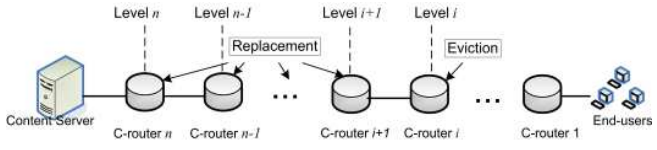


Fig. 3. An Example of Frequent Replacement Phenomenon of CLS.

a frequent replacement phenomenon. Fig. 3 depicts a path between a server and an edge C-router and shows an example of *frequent replacement* phenomenon.

When a content is evicted from C-router i at level i due to replacement (i.e., LRU algorithm), CLS forces to cache it at level i+1. If cache space of C-router i+1 is full, it will kick out another content and send it up to C-router i+2. If all the C-routers on the way to content server have full cache space, the content replacement and eviction will continue until arriving at content server. One eviction leads to continuous content replacement, which is defined as the "*Frequent Replacement phenomenon*".

IV. System Model

We consider an arbitrary network topology $G = (V, E)$ consisting of a set of N routers and a set of bidirectional links between these C-routers. C-router i ($i = 1, ..., N$) equipped with a CS module will cache the chunks of files and make content searching by exploiting in-network caching.

Each resource in the network is divided into small chunks* and an end-user requests the resource in the unit of chunk. For example, a resource that consists of 10 chunks will be requested by sending out 10 chunk requests from end-users. Also, each chunk is associated with an individual name (e.g., chunk-based naming [7]) and can be identified by C-routers. The chunk popularity varies across different domains and even the chunks of the same resource have different popularity: the forepart of a resource is requested more frequently, especially in video steaming.

In order to realize the chunk-level popularity-based content caching and searching, P-CLS creates a *Chunk-popularity Table* at each C-router to keep 3-tuple information pairs (*ChunkName; PopularityCount; TimeStamp*), indexed by a global chunk ID, the number of chunk counter and the time stamp when receiving a chunk from upstream or delivering it downstream. The overhead of *Chunk-popularity Table* has also been considered and can be greatly alleviated by adopting algorithms such as the message-digest (MD5) Hash algorithm [6]. Experiments [20] show that if a MD5 Hash value used 4 Bytes (i.e., 1B for *ChunkName* 2B for PopularityCount and 1B for TimeStamp), a C-router would need an additional 19.0735MB for containing one million chunks in the *Chunk-popularity Table*. This can be supported by the current network device techniques.

In content caching, collaborations among C-routers are required, in order to reduce chunk placement redundancy and avoid all C-routers caching the same set of chunks. In P-CLS, a C-router suggests caching a chunk downstream or evicting a chunk upstream by making the chunk with a cache suggestion Flag *CSF*. Such a Flag can be included in the Data packet header in CCN [7]. The value of *CSF* is set to 1 when a chunk is hit and modified to 0 after it is cached. When a chunk is evicted, the value of *CSF* is set to 2.

In content searching, traditionally a chunk request will be routed towards its original server. If the chunk request can not be satisfied by any C-router along the path, it will be forwarded towards the server. In P-CLS, if a C-router can not satisfy the chunk request, before sending it towards the server the C-router will try a downstream retrieval based on the *caching history trail*, helping to satisfy the request at a nearby C-router. The *caching history Trail* will be explained in next section.

*We use chunk and content interchangeably in this paper.

V. Popularity-driven P-CLS scheme

P-CLS scheme solves the tightly couple problem of content caching and searching in CCN through *Chunk-popularity Table* and *caching history Trail*.

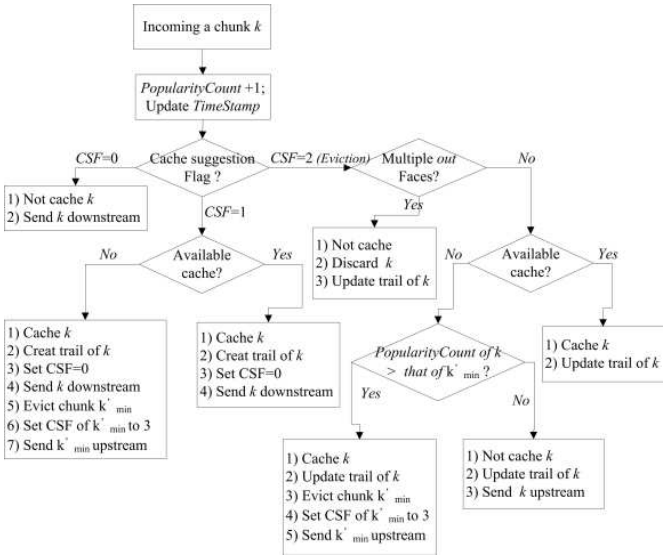


Fig. 4. Operations of a C-router in P-CLS when Receiving a Chunk.

A. Content Caching Operations

P-CLS dynamically makes the two main decisions of content caching: *where to cache* and *what to replace* based on chunk popularity.

1) ***Where to cache****:* A chunk can be cached by any C-router along the path between the hit node (the server or a C-router having the requested chunk) and the edge C-router. P-CLS guarantees there is at most one copy cached along its delivery path and dynamically adjusts the chunk caching location depending on the chunk popularity.

Let P_{ik} be the *PopularityCount* parameter of chunk k at C-router i. A hit chunk k at C-router i will be pulled down to

downstream C-router $i-1$ in two cases: (a) There is available cache space at C-router $i-1$ (regardless of chunk popularity); (b) Otherwise, $P_{(i-1)k} > P_{(i-1)k'_{min}}$ exits, where k'_{min} is the chunk that has the minimum *PopularityCount* value at C-router $i-1$.

P-CLS requires CCN Interest packet header to include a popularity comparison Flag *PCF* field. Every C-router refreshes the Flag *PCF* field of an unsatisfied Interest to indicate the next-hop C-router the popularity comparison relationship between the requested chunk k and the local least popular chunk k'_{min}. If $P_{(i-1)k} > P_{(i-1)k'_{min}}$, Flag *PCF* is set to be 1, otherwise 0. If the chunk request is hit at C-router i, the Flag *PCF* field will be extracted. If *PCF*$= 1$, C-router i marks C-router $i-1$ to cache the hit chunk (i.e., sets cache suggestion Flag *CSF* to 1) and deletes the hit chunk from its local CS. Note that Flag *CSF* will be reset to 0 at C-router $i-1$ to prevent the additional caching at downstream C-routers.

P-CLS guarantees that a chunk will not be pulled down to a C-router if its *PopularityCount* is less than the minimum one of the cached chunks. It is straightforward that C-routers would not want to cache a chunk at the cost of evicting a more popular one.

*2) **What to replace**:* When C-router $i-1$ receives a chunk k from upstream and its cache suggestion Flag *CSF* is 1, C-router $i-1$ will cache it directly if there is available space in CS; otherwise the chunk replacement will be carried out. In latter case, the chunk k'_{min} with the minimum value of *PopularityCount* in the cached chunks will be evicted. The evicted chunk k'_{min} with its Flag *CSF* set to 2 will be pushed back up by one hop.

Upon receiving an evicted chunk with *CSF*$= 2$, two cases exist in a C-router, which are shown in Fig. 5. Suppose that Fig. 5(a) is a random state of chunk caching and trails creation. Then an Interest for chunk G is sent from CR4 and hit at CR2. Depending on the *Where to cache* operations, CR4 decides to cache chunk G, has to evict chunk B and pushes it back up to CR2. Two cases of chunk replacement exist: (1) There is available space to cache chunk B at CR2, shown in Fig. 5(b). In this case, it is equivalent to simply swap the positions of chunk G and B. (2) The available space in CR2 because of pulling down chunk G has been occupied by a new coming chunk R, as shown in Fig. 5(c). Therefore, the arrival of chunk B may cause another chunk replacement. In this case, whether to replace chunk R for the new coming B depends on their *PopularityCount* values. The one which has larger *PopularityCount* value can be survived in the replacement operations. Fig. 4 describes the operations of a C-router in P-CLS when receiving a chunk.

B. Content Searching Operations

The content searching is based on the caching history trails created at C-routers along the chunk delivery path. Each trail is a 4-tuple entry (ID, *in*, *out*, *h*), including the information: (1) ID: a global unique chunk ID; (2) *in*: the Face of incoming C-router from which the chunk arrived; (3) *out*: the Face(s) of outgoing C-routers to which the chunk has been sent; (4) *h*: the number of hops from the server to this C-router.

The key difference of trails between P-CLS and CLS is that P-CLS trail considers arbitrary network topology and thus there may be multiple servers in the network. When the chunk arrives from a server, the value of in is not $null$ as that in CLS but is the Face of the server. For example, C-router CR1 connects to two servers and gets two chunk named B and R from server 1 and server 2, respectively. The trails created in P-CLS should be (B,Server1,*null*,1) and (R,Server2,*null*,1) but not as (B,*null*,*null*,1) and (R,*null*,*null*,1) in CLS. Explicit indication of chunk server in trails can avoid incorrect chunk being pulled back in chunk replacement operations.

The detailed content searching algorithm is shown in Alg. 1. C-routers will first check their CS and the Pending Interest Table (PIT) before using trails information to implement chunk retrieval. The lines 9-14 in Alg. 1 depict the generation of Flag *PCT* used for *where to cache* operation.

Algorithm 1 Chunk Searching Algorithm

```
1:  INPUT: Receive an Interest for chunk k.
2:  OUTPUT: Make request forwarding decision.
3:  if CS has not stored chunk k then
4:      Check the PIT;
5:      if PIT has a pending request for chunk k then
6:          Add the Interest arriving Face to the PIT record;
7:          Stop request forwarding.
8:      else
9:          Find the local least popular chunk k'_min;
10:         if PopularityCount of k'_min is smaller than that of k
            then
11:             Set PCF = 1 in the Interest packet header.
12:         else
13:             Set Flag PCF = 0.
14:         end if
15:         Check the trails.
16:         if A trail fits chunk k && its h > H_th  then
17:             Forward the Interest to a Face of out parameter.
18:         else
19:             Forward the Interest to the direction of server.
20:         end if
21:     end if
22: end if
```

VI. Performance Evaluation

In this section, we evaluate the proposed P-CLS scheme and compare it with existing LCE, LCD and CLS strategies as described in section 2. First, experimental evaluation is performed in a Java-built simulation environment for CCN. Second, a case study is given to illustrate how can P-CLS alleviate the *oscillation* and *frequent replacement* phenomenon of CLS.

A. Experimental Evaluation

The simulated topologies include *arbitrary* and *hierarchial* network topologies. All C-routers are equipped with CS mod-

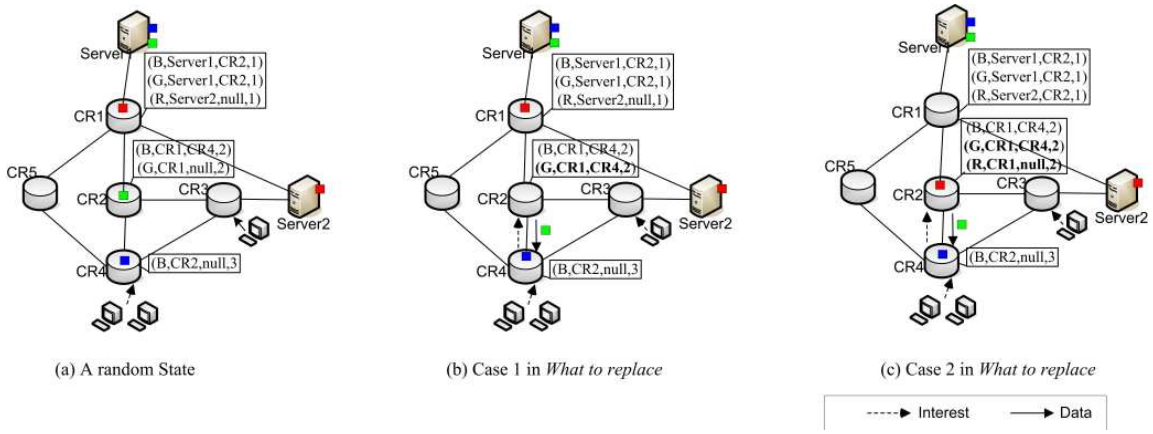


Fig. 5. An Example of *What to Replace* in P-CLS.

ules and the total size of these CSs is ranged from 10% to 50% of the whole content. C-routers are assigned with equal cache size for convenience. We consider 10000 resources and each resource is partitioned into chunks of unit size. Simulation launches 50000 request events following the Zipf distribution with a typical exponent $\alpha = 0.3$ and $\alpha = 0.9$. The bigger α indicates that few distinct content attract the majority of the requests while the smaller one indicates almost uniform content popularity.

Three metrics are used for performance evaluation, namely average cache hit ratio, path stretch and number of evictions caused by oscillation and frequent replacement phenomenon. We also exploit the impact of content popularity, cache size of CSs and network topology factors on the network performance.

Average Cache Hit Ratio: Fig. 6(a) and Fig. 6(c) depict the average hit ratio of requests curved as function of cache size for different caching schemes. The average hit ratio is defined as the ratio of the number of requests served by the C-router to the total number of requests arrived at the C-router.

- *Average Cache Hit Ratio vs. Total Cache Size.* With total cache size increases, the performance of average cache hit ratio becomes better. It is reasonable. More cache size of C-router can store more content, and thus preferentially requests can be satisfied by C-routers rather than the content server. The improvement of average cache hit ratio in CLS and P-CLS is not so obvious as that in LCE and LCD. The reason lies in that CLS and P-CLS take efforts to improve the content diversity of network. When cache size increases from 20% to 50%, the new coming content in CLS and P-CLS are those less popular content, which take limited impact on the network performance. As the total cache size increases from 30% to 50%, the performance of CLS is worse than LCD. The reason is related to the impact of network topology factor, which will be discussed later.
- *Average Cache Hit Ratio vs. Content Popularity.* Fig. 6(a) and Fig. 6(c) show the average hit ratio of requests in two types of content popularity ($\alpha = 0.3$ and $\alpha = 0.9$). All the evaluating schemes consistently perform better with a bigger α parameter. When α parameter increases from 0.3 to 0.9, the request arrival becomes more centralized, and thus C-routers can satisfy more requests with the same cache size.

P-CLS improves the average cache hit ratio over other schemes. This is not surprising because P-CLS can achieve content retrieval through chunk history trail and always caches the most popular chunk in C-routers.

Path Stretch: Fig. 6(b) and Fig. 6(d) depict the performance of path stretch. The path stretch is defined as $d/|P|$, where d is the number of hops that the data chunk has actually traveled, and $|P|$ is the path length from the requested user to the corresponding content server. Note that $d = 0$ when the content is cached at the edge access router and $d = |P|$ when the content is fetched from the content server, therefore $d/|P| \in [0, 1]$. A lower *path stretch* means less content download latency.

- *Path Stretch vs. Total Cache Size.* The impact of total cache size on the metric of path stretch is much better than that of average cache hit ratio. With total cache size increasing from 10% to 50%, the improvement of path stretch in P-CLS and CLS is not so obvious, especially in decentralized request arrival (i.e. $\alpha = 0.3$). When the total cache size is large enough (i.e. 50%) and the request arrival is relatively decentralized (i.e. $\alpha = 0.3$), CLS and P-CLS that keep at most one copy of content at their delivery path are not so effective in this case. With cache size large enough, LCE and LCD strategies can also cache the almost popular content as these content are frequently requested by end-users, and thus can also achieve a good performance. In practice, C-routers can rarely hold 50% of the whole Internet content.
- *Path Stretch vs. Content Popularity.* Fig. 6(b) and Fig. 6(d) show the performance of path stretch in two types of content popularity ($\alpha = 0.3$ and $\alpha = 0.9$). The benefit of P-CLS in path stretch metric becomes more

significant with α increasing. This implies that the more centralized the content popularity is, the more worthy performing the popularity-based chunk caching is.

Number of Evictions: Here we focus on the number of evictions that are caused by caching a new chunk at each C-router. Fig. 7 shows the performance comparison of the number of evictions between CLS and P-CLS. Because P-CLS always chooses to cache the more popular content in C-routers, it can reduce considerable number of unnecessary evictions. For example, P-CLS greatly reduces the number of evictions by 32% on average compared with CLS. It can also be seen that the number of evictions in P-CLS becomes smaller with α increasing from 0.3 to 0.9, but this rule can not found in CLS. This is because that CLS carries out chunk caching and evicting with LRU algorithm, without considering the content popularity factor.

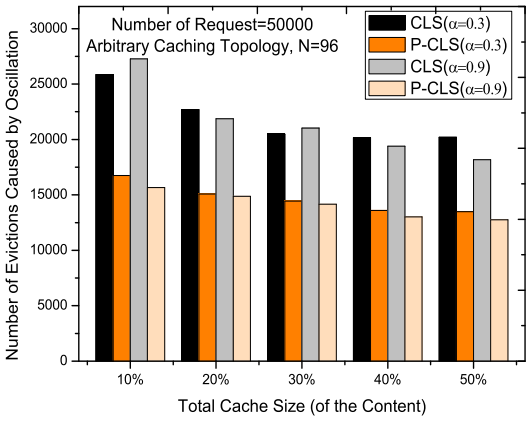


Fig. 7. Performance Comparison of Number of Evictions between CLS and P-CLS in Arbitrary Caching Topology (N=96).

Network Topology: Experiment results in Fig. 6 indicate that the performance of CLS is worse than LCD especially in large cache size. This conclusion is not consistent with related work [4], and the reason lies in the impact of network topology. Fig. 8 shows the average cache hit ratio comparisons in a hierarchical caching topology as that in [4]. It can be seen that CLS performs better than LCD in hierarchical caching topology regardless of cache size and content popularity factors. This is because that CLS is designed typically in hierarchical topologies and thus performs not so well in arbitrary network topologies. In a hierarchical topology, P-CLS is superior than CLS especially when the request arrival is more centralized (i.e., $\alpha = 0.9$).

B. A Case Study and Discussion

Here we use a case study to illustrate how P-CLS can alleviate the oscillation phenomenon, an example of which has been shown in Fig. 2.

Suppose that the *Chunk-popularity Table* of C-router 2 is (B, 20, t_B; R, 30, t_R; G, 10, t_G;). C-router 2 receives a request for chunk B and can not satisfy it, therefore sends it to C-router

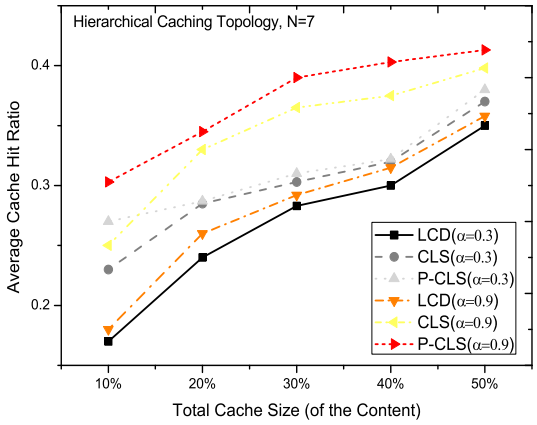


Fig. 8. Performance Comparison of Average Cache Hit Ratio between CLS and P-CLS in Hierarchical Caching Topology (N=7).

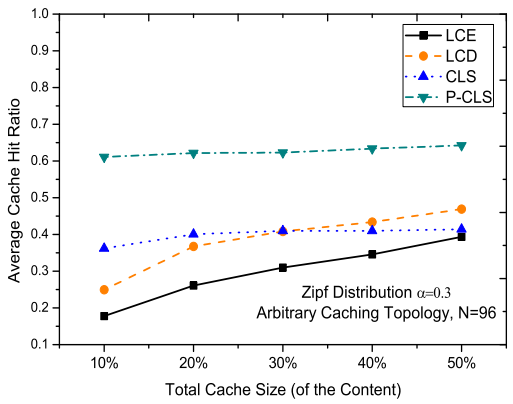
1 with the popularity comparison Flag $PCF = 0$. The request is hit in C-router 1. Because of $PCF = 0$, C-router 1 will not delete chunk B in local CS, but updates its chunk-popularity information and sends it downstream, marking C-router 2 not to cache it by setting the cache suggestion Flag $CSF = 0$. In this case, a cache hit in P-CLS does not cause 6 cache changes as that in CLS. Besides, a similar case study can also be given to show how can P-CLS alleviate the frequent replacement phenomenon of CLS by incorporating the *Chunk-popularity Table* into chunk placement and eviction.

P-CLS attempts to distribute content towards their (near) optimal positions by considering content popularity as well as the past content caching history trail. Sometimes an eviction will also cause several cache changes in P-CLS. Such cache changes are worthy as they are seeking an optimal chunk placement and will lead to whole network improvement.

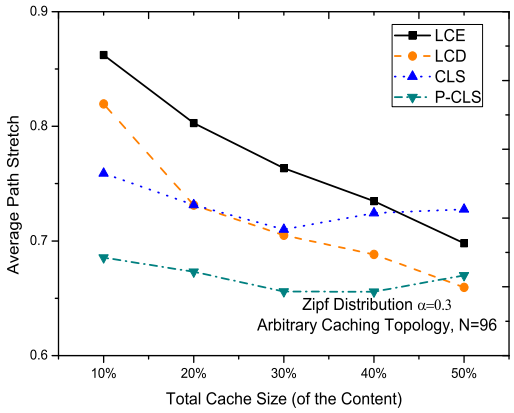
VII. Conclusion

This paper focuses on the tightly couple problem of content caching location and searching in CCN and proposes a popularity-driven content caching location and searching scheme, called P-CLS. P-CLS is designed in the arbitrary caching topology of CCN and works effectively to overcome the oscillation and frequent replacement phenomenon in CLS. Extensive simulations show that P-CLS is superior than the existing caching algorithms.

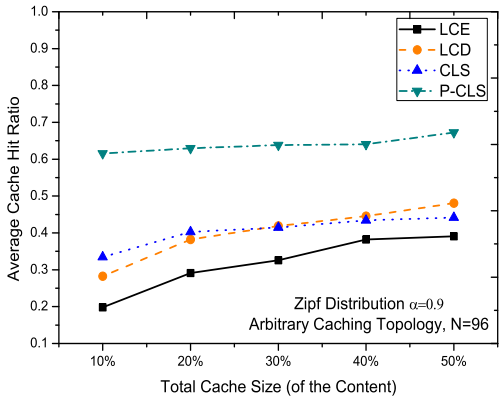
Our research also summarizes the following conclusions: (1) Total cache size, content popularity and network topology will affect the caching algorithms and the impact of content popularity is the most significant. The more centralized the request arrival is, the more benefit P-CLS can bring. (2) With the total cache size increasing, performance of all the caching algorithm improves, but potential of P-CLS decreases. (3) CLS is typically suit for hierarchial caching topologies and performs not so well in arbitrary topologies.



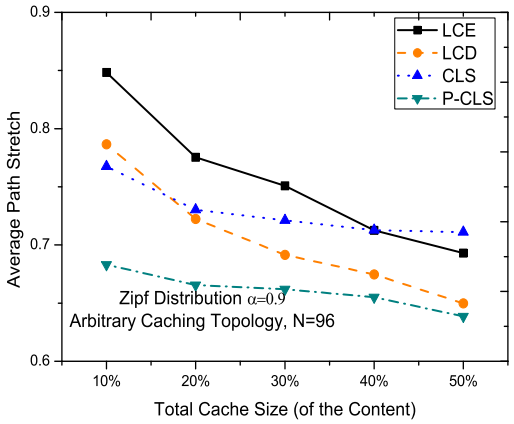
(a) Cache Hit Ratio vs. Total Cache Size ($\alpha = 0.3$).



(b) Path Stretch vs. Total Cache Size ($\alpha = 0.3$).



(c) Cache Hit Ratio vs. Total Cache Size ($\alpha = 0.9$).



(d) Path Stretch vs. Total Cache Size ($\alpha = 0.9$).

Fig. 6. Performance Comparisons between Different Caching Algorithms as a function of Total Cache Size ($\alpha = 0.3$ and $\alpha = 0.9$).

REFERENCES

[1] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an isp network," pp. 1200–1206, 2009.

[2] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *IEEE INFOCOM WKSHPS*, 2012, pp. 316–321.

[3] Cisco, "Cisco visual networking index: Forecast and methodology: 2011-2015," *White Paper*, 2012.

[4] Y. Li, T. Lin, H. Tang, and P. Sun, "A chunk caching location and searching scheme in content centric networking," in *ICC*. IEEE, 2012, pp. 2655–2659.

[5] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.

[6] D. Mau, M. Chen, T. Taleb, X. Wang, and V. Leung, "Fgpc: Fine-grained popularity-based caching design for content centric networking," pp. 295–302, 2014.

[7] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *ACM CoNEXT*, 2009.

[8] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," *Computer Communications*, vol. 25, no. 4, pp. 384–392, 2002.

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

[10] X. Tang and S. T. Chanson, "Coordinated en-route web caching," vol. 51, no. 6, pp. 595–607, 2002.

[11] Y. Xu, Y. Li, T. Lin, G. Zhang, Z. Wang, and S. Ci, "A dominating-set-based collaborative caching with request routing in content centric networking," in *IEEE ICC*, 2013.

[12] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.

[13] 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*, vol. 35, no. 12, pp. 3860–3884, 2008.

[14] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *IEEE INFOCOM*, 2012, pp. 2444–2452.

[15] A. Iosup, P. Garbacki, J. Pouwelse, and D. Epema, "Multiprobe project," URL: http://multiprobe.ewi.tudelft.nl, 2013.

[16] "Maxmind geoip database," URL: http://www.maxmind.com, 2013.

[17] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental evaluation of memory management in content-centric networking," in *IEEE ICC*, 2011, pp. 1–6.

[18] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown?: hamlet-like doubts in ICN," in *ACM ICN*, 2012, pp. 7–12.

[19] E. Rosensweig and J. Kurose, "Breadcrumbs: efficient, best-effort content location in cache networks," pp. 91–100, 2009.

[20] C. Bernardini, T. Silverston, and O. Festor, "Mpc: Popularity-based caching strategy for content centric networks," pp. 2212–2216, 2013.

[21] J. Li, H. Wu, B. Liu, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in named data networking," pp. 200–211, 2012.