Enhancing In-network Caching by Coupling Cache Placement, Replacement and Location

Xiaoyan Hu*, Jian Gong*, Guang Cheng*, Chengyu Fan†
*School of Computer Science and Engineering, Southeast University, Nanjing, P. R. China
†Computer Science Department, Colorado State University, Fort Collins, USA
Email: {xyhu, jgong, gcheng}@njnet.edu.cn chengyu@cs.colostate.edu

***Abstract*—As a distinctive feature of Information Centric Networking (ICN), in-network caching plays a fundamental role on system performance. The line-speed requirement of in-network caching invalidates the employ of complex collaborative caching schemes. The cache management of in-network caching includes three components – cache placement, cache replacement and cache location. Coupling the three pieces would make more efficient use of in-network caches, but existing in-network caching schemes consider only one or two of the three pieces. This work enhances in-network caching with a low complexity cache placement scheme that takes into account content popularity, hop reduction gains, cache space contention and replacement penalty and couples with cache replacement and location, here dubbed PRL (coupling cache Placement, Replacement and Location). PRL keeps data chunks that are more popular and farther away to fetch at an en-route router with less cache space contention. And PRL locates cached copies so as to serve a higher proportion of requests from in-network caches. Our preliminary simulation results suggest that PRL increases cache hit ratio and reduces the average hop count traversed by users' requests and the caching operations at routers as compared to existing representative in-network caching schemes.**

I. Introduction

According to Cisco's VNI report [1], it is anticipated that video traffic alone would represent 79% of all the IP traffic in 2018. That said, most of the traffic would be produced by content retrieval applications. To meet the growing demand of content delivery in a fundamental way, a clean-slate design, Information Centric Networking (ICN), such as Named Data Networking (NDN) [2], DONA [3] and PSIRP [4], focusing on named data rather than named host, is proposed. A distinctive feature of ICN infrastructure is that it enables intermediate routers[1] to cache passing by data chunks[2] so as to satisfy subsequent requests with cached copies. Such in-network caching plays a fundamental role on system performance.

An extremely large number of in-network caches are arranged as arbitrary networks and in-network caching requires search or replacement of data chunks at *line-speed* [5]. This line-speed requirement makes in-network caching fundamentally different from past overlay caching schemes, like Web caching [6], CDN and HTTP proxies in that it not only limits the cache size at each network level but also invalidates the employ of complex collaborative caching schemes in a centralized way [7], [8]. And thus in-network caching requires adjusting cache management operations to fit in a completely decentralized and little coordinated environment. Its cache management operations include cache placement (which picks content to cache at certain en-route router(s)), cache replacement (which determines what to replace) and cache location (which locates cached items). In most existing works on in-network caching, any en-route routers or a subset of traversed routers cache data chunks as they travel through the network. We believe that data chunks that are more popular and farther away to fetch should be kept and at an en-route router with less cache space contention. And cached copies should be located to serve a higher proportion of requests from in-network caches. However, as analyzed in section II, none of the existing works take such comprehensive consideration and couple cache placement, replacement and location. Consequently, the caching of an item determined by cache placement may evict a more valuable cached item and an item just inserted into a cache may be replaced before it serves any request. Furthermore, since content items cached in other routers are not located, the cached copies in neighboring routers are invisible to en-route routers and each en-route router can use merely content locally cached to serve users' requests.

This work is concerned with enhancing in-network caching with a low complexity cache placement scheme that takes into account content popularity, hop reduction gains, cache space contention and replacement penalty and couples with cache replacement and location, here dubbed PRL (coupling cache Placement, Replacement and Location). Our goal is to make more efficient use of available cache resources so as to potentially increase cache hit ratio and reduce the average hop count traversed by users' requests and the caching operations at routers. In PRL, upon the arrival of a request for a Data packet, a router computes the potential local caching contribution of the Data based on the local content popularity, hop reduction gain, cache space contention and replacement penalty; the request records the en-route router where the Data is with the maximal caching contribution; the returned Data would be cached at the en-route router specified in the request (where it is more popular, farther away to fetch and likely to stay longer) and leave trails in en-route routers near the caching router for them to locate the cached copy. Our preliminary simulation results corroborate that there is indeed a lot of space for resource management optimization of in-network caching policies, given that an appropriate combination of considerate cache placement, replacement and location is in place. The

[1]In this paper, the terms "router", "cache" and "node" are used interchangeable to refer to the network entity with caching capability.

[2]In this paper, the terms "Data packet", "data chunk", "content" and "content item" are used interchangeably to refer to a cacheable unit.

978-1-4673-6432-4/15/$31.00 ©2015 IEEE

main contribution of this work is threefold:

- We propose an implicit coordinate chunk placement scheme that comprehensively considers content popularity, hop reduction gains, cache space contention at en-route routers and replacement penalty.
- We tie the cache placement, replacement and location together. The cache replacement chooses a cached item with the least caching value to evict. The cache location piece creates a trail for a passing item that would be cached at a nearby downstream router as determined by the cache placement to direct future requests to the cached copy.
- We exhibit the effectiveness of the PRL scheme by extensive ndnSIM based simulations and compare PRL with several well-known in-network caching schemes.

II. Related Work

There is a large body of literature on in-network caching. The default in-network caching placement in ICN is Caching Everything Everywhere (CEE) [2], i.e. every en-route router caches every Data packet passing by, resulting in cache redundancy and unnecessarily frequent cache replacement. Then the cache placement has spawned interest in topics such as caching redundancy reduction [9], [10], [11] and caching prioritization by popularity assessment [12], [13]. The caching schemes in [9], [10], [11] do not discriminate content items resulting in unnecessary cache replacement. A node's caching decision in the caching scheme of [12] is made by another node and based on the popularity information at that node, which may generate unnecessary duplicates. SAC [13] incurs network cache resource waste as a less popular content cached at an upper layer is repeatedly pulled down and pushed back by following requests. Besides popularity, MAGIC [14] and OPPORTUNISTIC [15] consider hop reduction gains and CCCCN [16] takes cache space contention at en-route routers into account, but none of them consider the three factors together.

Moreover, existing works on the cache placement make caching decisions without consideration of cache replacement penalty (except MAGIC), which may lead to impertinent caching and replacement operations. And they run with one of the two most popular cache replacement strategies in ICN, Least Recently Used (LRU) and Least Frequently Used (LFU) [2]. LRU and LFU prioritize cached chunks based on only their recency or popularity, while Least Benefit (LB) [17] improves LFU by considering hop reduction gains. However, the work in [17] discusses nothing about what kind of cache placement strategy should combine with LB.

To utilize in-network caches in an efficient way, it is crucial to make temporary cached content to be visible in its vicinity by cache location. The works in [18], [19] propose to aggressively advertise the reachability of cached content in the routing system, but it is impractical as it imposes significant burden on the routing system in terms of maintaining dynamic states of a large number of high volatile cached items. Instead, Breadcrumbs [20] adopts an implicit and best effort approach towards cache location by storing the most recent direction and time that an item was forwarded. However, Breadcrumbs works with the cache placement of CEE and creates breadcrumbs for each delivered item at each en-route router.

We argue that cache placement scheme should comprehensively consider content popularity, hop reduction gains, cache space contention at en-route routers and replacement penalty and tie with cache replacement and location. We believe that data chunks that are more popular and farther away to fetch should be cached at an en-route router with less cache space contention and cached copies should be located, which would serve a higher proportion of requests from in-network caches and reduce the average hop count traversed by users' requests and the caching operations at routers. This is the basic idea of our PRL and corroborated by our preliminary simulation results.

III. Building PRL

In this section, we first state the system model and then elaborate on the in-network caching PRL. Note that PRL can be applied to almost all ICN architectures that enable in-network caching. But to ease the description, in the following sections, we consider the specific in-network caching problem in NDN [2], the promising paradigm of ICN.

A. System Model

We consider an NDN network of an arbitrary topology represented by a graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ denotes the set of routers and E the communication links interconnecting them. Each router (i.e., in-network cache) is with caching capability and a limited cache capacity. Let $O = \{o_1, o_2, \ldots, o_{|O|}\}$ represent the set of Data packets requested by users and hosted by certain content servers in the network. Without loss of generality, we assume that Data packets are of the same size and each cache slot in a content store can accommodate one Data packet. We assume that requests arrive in the network exogenously as a Poisson process. While NDN naturally supports multi-path forwarding to enhance network performance, it is still a non-deterministic variation which depends upon the development of the future protocol. For simplicity, we restrict content caching to the en-route principle. Namely, an Interest is first routed towards relevant content server via the shortest path unless redirected by a trail in an en-route router. The matching Data packet returns via the reverse path and may be cached at one of the routers that it passes by.

TABLE I: MODEL NOTATION

Symbol	Meaning
R	The sequence of requests sent by users
S_v	The cache capacity of node $v \in V$ in terms of the number of content units
C_v	The set of content cached at node $v \in V$ and $C_v \subset O$
e_v	The average number of caching eviction operations occurred per unit time at node $v \in V$
r_v^o	The request rate of $o \in O$ at node $v \in V$
h_v^o	The hop count from node $v \in V$ to the source of $o \in O$, or to say the hop reduction gain of o if it is cached at v

In an NDN router, while the same Interests are almost simultaneously received from several users, only the first arrival would be sent upstream for exploring the desired data. In other words, the requests for the same data from different users are aggregated at downstream routers and only one Interest is seen by upstream routers (i.e., requests aggregation). Furthermore,

requests that get cache hits at downstream routers cannot be seen by upstream routers either, viz., cache filtering effect [21]. Hence, nodes in the network have distinct views of content popularity distribution. In our system, each router periodically obtains local content popularity distribution information by statistics from users' data access history which reflects the requests aggregation and cache filtering effect.

Due to the difference in cache size or traffic load or cache management technology, cache space contentions at different en-route routers are diverse. And thus the time that a cached copy can stay in different routers differs. And an item should be placed at an en-route router that tends to keep it longer so that it has more opportunity to serve future requests. A straightforward metric to measure the cache space contention at a router $v \in V$ is e_v, its average number of caching evictions per unit time. The larger e_v is, the more cache space contention at router v and the sooner its cached items are likely to be evicted.

A cache hit is recorded for a request if it finds a matching item in an en-route router or a neighboring router of an en-route router. Alternatively, a cache miss is recorded and the Interest traverses the full delivery path to the content server. Upon the arrival of an uncached chunk, a router determines whether to cache the newcomer based on our proposed cache placement. If yes and its content store is full, a cached Data packet is replaced according to the replacement policy. otherwise, if the router determines no caching and realizes that a downstream router to which it is nearer than to the content source would cache the item, its local cache location component creates a trail entry to record such caching for redirecting future requests downstream. To facilitate further discussion, related symbols are explained in Table I.

B. The Proposed PRL

As mentioned above, PRL couples cache placement, replacement and location. We present the three pieces separately in the following subsections.

1) ***Cache Placement Policy****:* The cache placement determines if a Data packet should be cached and where it should be placed on its delivery path based on the caching contributions of the Data at these en-route routers. We first define the caching value of an item $o \in O$ which could then be cached or is cached at a node $v \in V$ as follow:

$$Value_v^o = \frac{r_v^o \times h_v^o}{e_v} \tag{1}$$

It measures the local popularity (r_v^o) and hop reduction gain (h_v^o) of item o and the cache space contention (e_v) at router v. If an Interest for $o \in O$ arrives at a router $v \in V$ and o is not locally cached, v would estimate the potential caching contribution of o which relates to if the cache space of v is fully occupied and is defined as follow:

$$Contri_v^o = \max\{0, Value_v^o - RepPenalty_v\} \tag{2}$$

where $RepPenalty_v$ is defined as

$$RepPenalty_v = \begin{cases} 0 & \text{if } |C_v| < S_v, \\ \min\{Value_v^k | k \in C_v\} & \text{if } |C_v| = S_v. \end{cases} \tag{3}$$

More specifically, if the local cache is not full yet, the caching of o at v would incur no caching evictions and the local caching contribution of o would be exactly its local caching value. Otherwise, the caching of o would incur the eviction of certain cached item and such eviction leads to replacement penalty since future requests for the evicted item cannot be served by the router any more. And thus, in such case, the local caching contribution of o is its local caching value minus the replacement penalty.

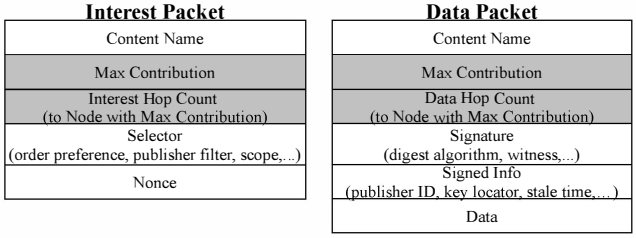


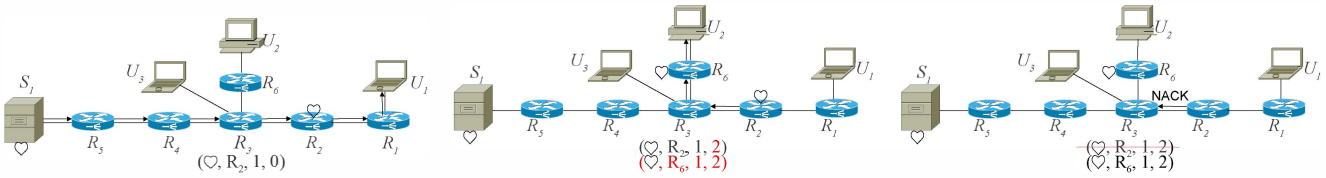
Fig. 1: Modified NDN Interest and Data packets.

Figure 1 illustrates the modified NDN Interest and Data packets for PRL where the shadowed fields are these added for PRL. During the journey from its requester to its content source, an Interest records its maximal caching contribution at these passing routers in its *Max Contribution* field. Its *Max Contribution* value is then copied to the *Max Contribution* field of its matching Data packet. Then the returned Data packet would be cached at the router where the Data has the maximal caching contribution. *Note that a Data packet would be cached on its delivery path only if its maximal caching contribution is positive.*

2) ***Cache Replacement Policy****:* The cache replacement policy prioritizes cached content items not only by their request rates, but also by their hop reduction gains. Whenever a Data packet is going to be inserted into a full cache, the local cache replacement policy would choose the one with minimal caching value among all the cached content to replace. In this way, the cache replacement policy is tightly coupled with the cache placement policy. Its replacement leads to the maximal contribution of the new caching as planned by the cache placement algorithm.

According to the definition of caching value in equation 1, at a specific router $v \in V$, the cache replacement policy can prioritize its cached content item $o \in O$ by r_v^o and h_v^o without consideration of e_v since e_v can be assumed to be the same for all the items cached at v. The calculation of local caching value of a cached item $o \in O$ at $v \in V$, i.e., $r_v^o \times h_v^o$, and the priorization of cached content items can be implemented as follow similar as that in LB [17]: whenever o serves a request, its local caching value is increased by its hop reduction gain and then o is ranked based on its resulting caching value.

3) ***Cache Location Policy****:* The cache location component at a router maintains trails for passing content items. Such trails locate content items cached downstream so as to serve future requests from nearby in-network caches. Note that an en-route router creates a trail for a passing item only if the item is not locally cached but cached at a downstream router to which it is nearer than to the original content source.



(a) Trail Creation (b) Request rerouting with trails (c) Deprecated trail

Fig. 2: An example of cache location according to the trails.

Each trail is a 4-tuple entry $(content\_name, next\_hop, hop\_count, latest\_time)$ indexed by the content name of the cached item, the next hop and hop count to the caching point, and the most recent time that the trail is referred to. The most recent time is initialized as the time the trail is created and updated whenever the trail is used to successfully fetch data. The *Interest Hop Count* and *Data Hop Count* fields of Interest and Data packets shown in Figure 1 are maintained by en-route routers. From the *Data Hop Count* value, en-route routers know an upstream or downstream router would cache the Data and the distance to the caching point. The *Interest Hop Count* of an Interest packet is initialized as 0 and increased by 1 at each en-route router, but reset to 0 whenever the *Max Contribution* of the Interest is updated. The *Data Hop Count* of a Data packet is first copied from the *Interest Hop Count* of the requesting Interest and then decreased by 1 at each en-route router. The value of *Data Hop Count* can be positive or negative which indicates the caching point is a downstream or upstream router.

Figure 2 shows an example of how a trail is created, used and deprecated. As illustrated in Figure 2(a), the request for ♡ from U_1 arrives at the content server S_1 via the path $R_1 - R_2 - R_3 - R_4 - R_5$ and ♡ is determined to be cached at R_2. When ♡ traverses the symmetric path to return to U_1, each en-route router knows the direction of the caching point, upstream or downstream, and its distance to the caching point. Only R_3 creates a trail for ♡ as it realizes ♡ will be cached at its downstream router R_2 which is one hop away and from which it is nearer to fetch ♡ than from the content server S_1. Subsequently, as shown in Figure 2(b), the request for ♡ from U_2 is first routed towards S_1 and then meets R_3 which maintains a trail for ♡ and redirects the request to the cached copy at R_2. As the cached copy of ♡ returns to U_2, R_3 updates the $latest\_time$ of the forwarding trail and creates another trail for ♡ since ♡ would be cached at its downstream router R_6 which is just one hop away. Expired trails that are not in active service for 10 seconds [10] (configurable) would be purged periodically. If a cached chunk is replaced at its caching point, the trails that keep track of the caching become invalid. Such an invalid trail may be used to direct Interest forwarding before its purge. As displayed in Figure 2(c) where the request for ♡ from U_3 is directed to R_2 by the invalid trail at R_3 after R_2 evicts the cached copy, a NACK [22] would be sent back by the original caching point R_2 so that the invalid trail at R_3 is deprecated as soon as possible.

PRL is a lightweight scheme that does not require explicit information exchange among in-network caches and each router makes cache placement, replacement and location decisions by leveraging its own information. Each router does popularity statistics and maintains trail entries for PRL. The memory consumption for popularity statistics at a router is proportional to N, the number of distinct objects accessed by the router during the statistical interval. As NDN content names are with variable length, popularity statistics can store the hash values of these names instead to reduce space usage. Moreover, space efficient double counting Bloom filter [23] can be used for recording content popularity to further reduce space usage, which is similar as that in [23] for making statistics of flow length on high speed networks. Note that only content items with positive caching contributions are cached and in at most one en-route router, and an en-route router creates a trail for a passing item only if the item is not locally cached but cached at a downstream router to which it is nearer than to the original content source. And therefore the number of trails that a router maintains should not be large and the memory for storing trail entries should not be significant.

IV. Performance Evaluation

In this section, we present an in-depth evaluation to quantify the effectiveness of our in-network caching scheme and also evaluate the factors that impact its performance.

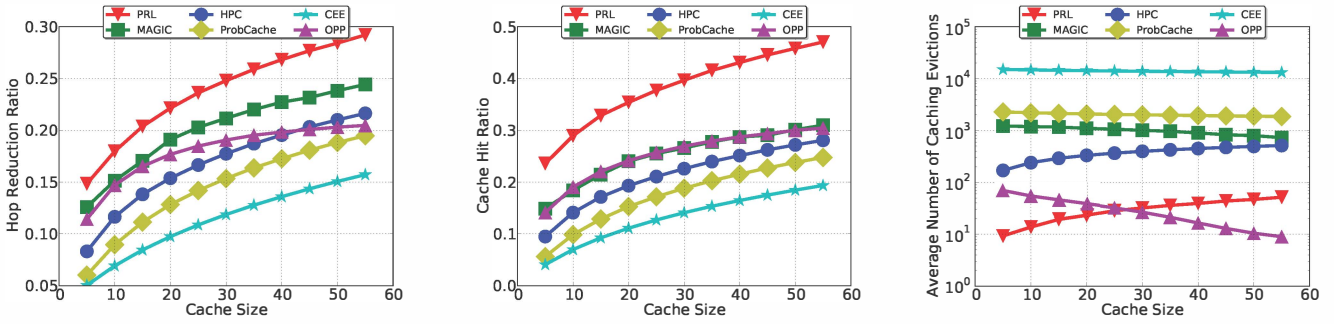
A. Experimental Setup

We use the open-source ndnSIM [24] package which implements the NDN protocol stack for the NS-3 network simulator(http://www.nsnam.org/) to run simulations for a variety of scenarios on a 2.70GHZ CPU with RAM 2.0GB. We extend ndnSIM by adding *Max Contribution* and *Interest Hop Count* fields in Interest packets and *Max Contribution* and *Data Hop Count* fields in Data packets , and customizing the forwarding strategy and replacement policy to perform cache placement, replacement and location with our proposed in-network caching PRL.

Benchmark. We compare our proposed scheme (**PRL**) against other representative in-network caching schemes:

- **MAGIC**, the max-gain in-network caching [14],
- **HPC**, the hop-based probabilistic caching [11],
- **OPP**, the opportunistic in-network caching [15],
- **ProbCache**, the probabilistic caching scheme based on path-capacity [10],
- **CEE**, the default in-network caching in which content items are cached at all the en-route nodes [2].

We configure the replacement policies of different benchmarks according to their corresponding literatures. For MAGIC, LFU



(a) Hop reduction ratio β (b) Cache hit ratio γ (c) Average number of caching evictions δ

Fig. 3: Caching performances versus cache sizes ($\alpha = 0.8$, $D = 5$)

.

is used as its replacement policy. Instead, for HPC, OPP, ProbCache and CEE, LRU is used.

Network Topology. Our simulations are conducted in a non-complete K-ary tree topology. We assume that the root node is the content server and all requests are sent from the leaf nodes, similar to the setting of [14]. A tree is small enough to be amenable to such analysis and instructive since from the view of a content server, the content distribution topology is effectively a tree. For a K-ary tree topology, there are two parameters: k which denotes the maximum number of children of each node and D which represents the depth of the tree. Similar as that in [14], $k = 5$ is used in our simulations in which the number of children for each node is randomly generated in the range of $[0, 5]$.

TABLE II: PARAMETER SETTING FOR SIMULATIONS

Parameter	Default Value	Range
request rate λ	$100 interests/s$	
the number of requests $\|R\|$	5×10^5	
the size of requested data set $\|O\|$	1×10^4	
tree ary k	5	
tree depth D	5	$[3, 7]$
node cache capacity S_v	20	$[5, 55]$
Zipf shape parameter α	0.8	$[0.7, 1.4]$

Methodology. We evaluate how cache size, users' access pattern and the depth of K-ary tree impact the performance of our in-network caching PRL. For each test run, we assume that users express Interests as a Poisson process and generate 5×10^5 requests to allow the system to reach steady states. The number of distinct content units in the network is 1×10^4 and users' access of these items follows the Zipf distribution. We set homogeneous cache sizes at all routers as Rossi et al. [25] proved that the gain brought by content store size heterogeneity is limited. Each link in our considered topologies is assigned a bandwidth of $10 Gbps$ greater than the traffic demand and a propagation delay of $1 ms$. Table II shows the setting of primary parameters in our simulations.

Performance Metrics. We quantify the performance of the in-network caching strategies from the following three aspects:

- **Hop reduction ratio**, which indicates the improvement in user-perceived data delivery quality and is defined as

$$\beta = 1 - \frac{\sum_{r \in R} h_r}{\sum_{r \in R} H_r} \tag{4}$$

where h_r and H_r are the hop counts from the requester of $r \in R$ to the node which serves the request and to the original content server respectively.

- **Cache hit ratio**, which demonstrates the reduction in server load and distribution load of all the transient networks along the paths from the content server to end users due to in-network cache hits and is defined as

$$\gamma = \frac{w}{|R|} \tag{5}$$

where w is the total number of requests that are satisfied by in-network caches.

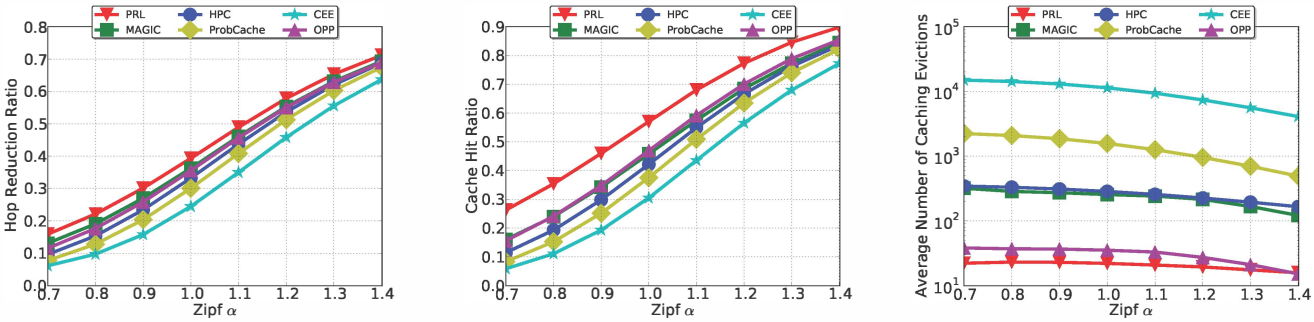
- **Average number of caching evictions** occurred at each node during a simulation run, which reflects the time and power consumption in I/O operations of cache storage and is defined as

$$\delta = \frac{\sum_{v \in V} E_v}{|V|} \tag{6}$$

where E_v is the total number of caching evictions at node $v \in V$ during a simulation run.

B. Results and Discussion

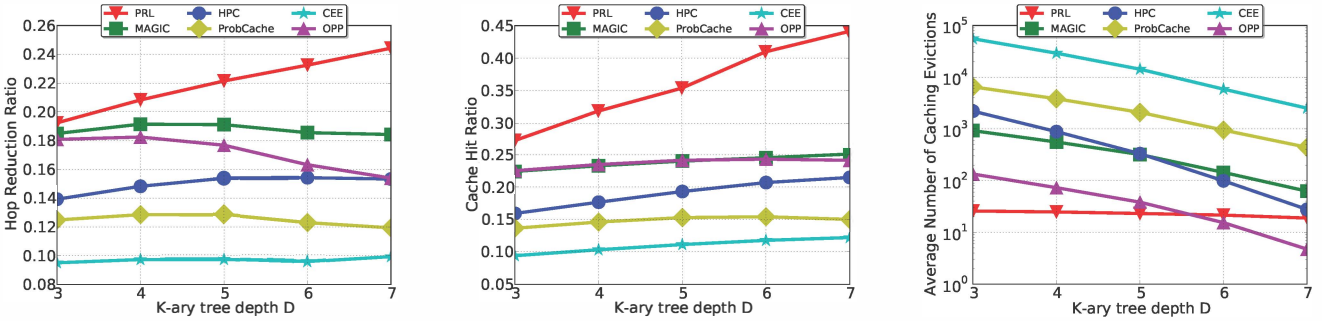
*1) **The impact of cache size**:* Figure 3 demonstrates how cache size impacts the caching performance of the considered in-network caching schemes. As illustrated, PRL performs the best among all the considered caching schemes in terms of hop reduction ratio and cache hit ratio (an improvement of around $15.88\% \sim 19.58\%$ and $47.37\% \sim 58.54\%$ respectively over the second best strategy, the MAGIC). And the improvements increase as cache size increases since more content items can be held in caches and then more requests can be served by in-network caches and traverse less hops. For the average number of caching evictions, PRL reduces it to the smallest among the evaluated caching schemes (a reduction of around $11.99\% \sim 86.50\%$ over OPP, the one generated the second smallest) until the cache size increases to 30 where OPP starts to incur the least evictions. And the average number



(a) Hop reduction ratio β (b) Cache hit ratio γ (c) Average number of caching evictions δ

Fig. 4: Caching performances versus Zipf shape α ($D = 5$, $C_v = 20$ for $v \in V$)

.



(a) Hop reduction ratio β (b) Cache hit ratio γ (c) Average number of caching evictions δ

Fig. 5: Caching performances versus tree depth D ($\alpha = 0.8$, $C_v = 20$ for $v \in V$)

.

of caching evictions generated by PRL presents a trend of increase as the cache size increases. The reason is that under PRL, when cache space is not fully occupied, content items are cached without consideration of replacement penalty. While the cache space is filled up, PRL reconsiders the elects based on the replacement penalty of the full cache and the number of reelections (i.e., evictions) is proportional to the cache size. Since PRL considers content popularity, hop reduction gains, cache space contention and replacement penalty when making reelection decisions, content items are picked to cache at appropriate locations and seldom evictions occur after the reelection. And thus the number of caching evictions incurred by PRL is small.

*2) **The impact of Zipf shape parameter** α:* Figure 4 illustrates how the Zipf shape parameter α impacts the caching performance of the considered in-network caching schemes. It can be seen that PRL outperforms the other caching schemes considered herein anyway in terms of the three performance metrics. As α increases, both the hop reduction ratio and cache hit ratio of all the considered caching schemes increase, while the average number of caching evictions decreases. The reason is that the popularity of hot content increases as α increases, and then caching these most popular content items results in more cache hits, requests traversing less hops and less cache replacement. As shown, the improvements of PRL in terms of hop reduction ratio and cache hit ratio compared to MAGIC, the second-best, are around $2.65\% \sim 21.42\%$ and $6.42\% \sim 64.02\%$ respectively. For caching evictions, PRL generates the least among all the caching schemes considered herein (a reduction of around $17.01\% \sim 41.86\%$ compared to OPP, the one incurred the second least) until that when $\alpha = 1.4$, OPP generates even smaller number of caching evictions.

*3) **The impact of tree depth**:* Figure 5 shows how the tree depth D impacts the caching performance of the considered in-network caching schemes. As expected, PRL still stands out with different settings of the tree depth D. For the hop reduction ratio and cache hit ratio, PRL outperforms the other caching schemes considered herein and its improvements increase as the tree depth D increases. This is because in our simulations, a larger tree depth means a larger number of nodes. That is, a larger cache space for PRL to optimize content caching and location for serving users' requests. Its improvements in hop reduction ratio and cache hit ratio compared to MAGIC, the second-best, are around $4\% \sim 32.61\%$ and $21.94\% \sim 75.76\%$ respectively. For the average number of caching evictions, PRL incurs only dozens of times. When $3 \leqslant D \leqslant 5$, PRL generates the smallest among all the caching schemes considered herein (a reduction of around $39.54\% \sim 80.72\%$ compared to OPP, the one incurred the second smallest); and when $6 \leqslant D \leqslant 7$, OPP generates even smaller number of caching evictions. Furthermore, the average

number of caching evictions of all the considered caching schemes decreases as the tree depth D increases. The reason is that as the tree depth D increases, the total cache space increases, which usually leads to less replacement. But due to the line-speed requirement, we expect that the cache sizes of NDN routers in the future should not be very large and the total cache space relative to the large number of content items in the network is small. And the above simulation results demonstrate that PRL performs well in such cases.

V. Conclusion & Future Work

This work is concerned with enhancing in-network caching with a low complexity cache placement scheme that considerately takes into account content popularity, hop reduction gains, cache space contention and replacement penalty and couples with cache replacement and location, here dubbed PRL. PRL keeps data chunks that are more popular and farther away to fetch at an en-route router with less cache space contention. And PRL locates cached copies so as to serve a higher proportion of requests from in-network caches. Our preliminary results of ndnSIM based simulations corroborate that PRL makes more efficient use of available cache resources. It potentially increases cache hit ratio by up to 75.76%, and reduces the average hop count traversed by users' requests and the average number of caching evictions occurred at routers, a reduction of up to 32.61% and 86.50% respectively. Our next step is to further evaluate our PRL in a more realistic topology with trace-driven requests, refine each piece of the system and then deploy it in Named Data Networking (NDN) testbed.

Acknowledgment

This work was conducted under the support of Jiangsu Provincial Key Laboratory of Computer Network Technology and Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, and sponsored by the National Grand Fundamental Research 973 program of China under Grant No.2009CB320505, the National Nature Science Foundation of China under Grant No. 60973123, the Technology Support Program (Industry) of Jiangsu under Grant No.BE2011173, and the Prospective Research Project on Future Networks of Jiangsu Future Networks Innovation Institute under Grant No.BY2013095-5-03. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of those sponsors.

References

[1] Cisco, "Cisco visual networking index: forecast and methodology: 2013-2018," June 2014.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. CoNEXT'09*, 2009, pp. 1–12.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, August 2007.

[4] M. Särelä, T. Rinta-aho, and S. Tarkoma, "RTFM: Publish/subscribe internetworking architecture." ICT Mobile Summit, 2008.

[5] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *Proc. ReARCH '10*, 2010, pp. 5:1–5:6.

[6] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of Web caching architectures: hierarchical and distributed caching," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 404–418, 08 2001.

[7] P. Sarkar and J. H. Hartman, "Hint-based cooperative caching," *ACM Trans. Comput. Syst.*, vol. 18, no. 4, pp. 387–419, 2000.

[8] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in named data networking," in *Proc. ANCS'12*, Austin, TX, USA, 2012, pp. 15–26.

[9] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *Proc. IFIP'12*, 2012, pp. 27–40.

[10] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. ICN'12*, 2012, pp. 55–60.

[11] Y. Wang, M. Xu, and Z. Feng, "Hop-based probabilistic caching for information-centric networks," in *Proc. GLOBECOM*, 2013, pp. 2102–2107.

[12] 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 *Proc. NOMEN'12*, Orlando, Florida, USA, March 2012, pp. 316–321.

[13] Y. Li, Y. Xu, T. Lin, G. Zhang, Y. Liu, and S. Ci, "Self assembly caching with dynamic request routing for information-centric networking," in *Proc. GLOBECOM*, 2013, pp. 2158–2163.

[14] J. Ren, W. Qi, C. Westphal, J. Wang, K. Lu, S. Liu, and S. Wang, "Magic: A distributed max-gain in-network caching strategy in information-centric networks," in *Proc. INFOCOM Workshops NOM*, 2014, pp. 470–475.

[15] X. Hu and J. Gong, "Opportunistic on-path caching for named data networking," *IEICE Transactions on Communications*, to be published in Nov 2014.

[16] S.-W. Lee, D. Kim, Y.-B. Ko, J. Kim, and M.-W. Jang, "Cache capacity-aware ccn: Selective caching and cache-aware routing," in *Proc. GLOBECOM*, 2013, pp. 2114–2119.

[17] S. Wang, J. Bi, and J. Wu, "On Performance of Cache Policy in Information-Centric Networking," in *Proc. ICCCN'12*, 2012, pp. 1–7.

[18] Y. Wang, K. Lee, B. Venkataraman, R. L. Shamanna, I. Rhee, and S. Yang, "Advertising cached contents in the control plane: Necessity and feasibility." in *Proc. NOMEN'12*, 2012, pp. 286–291.

[19] W. Wong, L. Wang, and J. Kangasharju, "Neighborhood search and admission control in cooperative caching networks," in *Proc. GLOBECOM*, 2012, pp. 2852–2858.

[20] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks." in *Proc. INFOCOM'10*. IEEE, 2009, pp. 2631–2635.

[21] C. L. Williamson, "On filter effects in web caching hierarchies." *ACM Trans. Internet Techn.*, vol. 2, no. 1, pp. 47–77, 2002.

[22] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, Jun. 2012.

[23] H. Wu, J. Gong, and W. Yang, "Algorithm based on double counter bloom filter for large flows identification," *Journal of Software*, vol. 21, no. 5, pp. 1115–1126, 2010.

[24] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN, Technical Report NDN-0005, October 2012.

[25] D. Rossi and G. Rossini, "On sizing ccn content stores by exploiting topological information," in *Proc. NOMEN'12*, Orlando, Florida, USA, March 2012, pp. 280–285.