IEICE TRANSACTIONS

on Communications

VOL. E97-B NO. 11
NOVEMBER 2014

A PUBLICATION OF THE COMMUNICATIONS SOCIETY

The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

IEICE TRANS. COMMUN., VOL.E97–B, NO.11 NOVEMBER 2014

PAPER

Opportunistic On-Path Caching for Named Data Networking

Xiaoyan HU[†,††a)], ***Student Member and*** **Jian GONG**[†,††b)], ***Nonmember***

SUMMARY As a prominent feature of *Named Data Networking* (NDN), in-network caching plays an important role in improving the performance of content delivery. However, if each NDN router indiscriminately caches every data packet passing by (i.e., *Caching Everything Everywhere* (CEE)), the result can be unnecessarily frequent cache replacement and cache redundancy in en-route routers and thus in-network caches are not utilized in an efficient way [1], [2]. Moreover, managing these in-network caches in a centralized way may lead to excessive resource consumption since the number of these caches is considerable. This work proposes a distributed and opportunistic on-path caching scheme. To be specific, each en-route router independently picks content items to cache in such a way that popular content is more likely to be cached by routers, especially routers near users, and cache redundancy is reduced. Extensive simulations including trace-driven ones in a PoP-level ISP topology suggest that the proposed scheme improves the average cache hit ratio of users' requests and reduces the average hop count as compared to CEE and the other on-path caching algorithms considered herein.

key words: *NDN, On-path caching, popularity, hop count*

1. Introduction

In *Named Data Networking* (NDN) [3], the promising paradigm of *Information Centric Networking*(ICN) [4], in-network caching plays an important role in improving the performance of content delivery. However, if NDN routers* indiscriminately cache any data packets** passing by, i.e., *Caching Everything Everywhere* (CEE), the result can be unnecessarily frequent cache replacement and redundant caching in these routers, which attenuates the effectiveness of in-network caching [1], [2]. More particularly, in CEE, any router that receives a solicited data packet caches it. If the cache is full, sufficient cached data packets are selected and dropped to allow insertion of the new arrival. Then frequent replacement is generated as data packets arrive at routers and redundant copies are held in en-route routers. Furthermore, as the request popularity of content usually follows a Zipf-like distribution [5], an extremely large number of objects in the network may be accessed by users with fairly small frequencies. In CEE, such objects with fairy small access frequencies may replace relatively popular content cached in en-route routers and then their redundant copies in en-route routers may be replaced before they can serve any future requests, whereas the requests arriving immediately for the replaced popular objects cannot get cache hits any more.

To improve the effectiveness of in-network caching, cache management has attracted wide attention in the ICN research community. There is a large body of literature on collaborative caching [6]–[8] (off-path caching) in traditional Web caching. However, as the number of in-network caches is considerable, collaborative caching in NDN, if not well designed, could significantly increase the communication overhead among nodes and its yield may be marginal. In contrast, on-path caching requires less coordination among caches and data packets are cached by any en-route routers or a subset of traversed routers as they travel through the network. On-path caching has spawned interest in topics such as reducing caching redundancy [1], [2] and caching prioritization by popularity assessment [9], [10]. The caching schemes in [1], [2] do not discriminate content items resulting in unnecessary content replacement. A node's caching decision in the caching schemes of both [9] and [10] is made by another node and is based on the popularity information at that node, which may generate unnecessary duplicates.

This work aims to reduce upstream bandwidth demand and improve data delivery performance of an NDN domain by proposing a *distributed* and *opportunistic* on-path caching scheme. The idea behind our opportunistic on-path caching is that more popular content would serve a higher proportion of the total requests and caching content near users would reduce the average number of hops traversed by users' requests. In this scheme, each en-route router independently decides the probability of caching a specific data packet based on the request popularity of the data observed by the router and the distances from the router or the content server of the data to the requester. As a result, popular content is more likely to be cached by routers, especially routers near users. Such opportunistic on-path caching is a lightweight scheme without requiring explicit information exchange among caches. It reduces cache space contention and cache redundancy in en-route routers and keeps popular content for longer in caches near users. Extensive simulations including trace-driven ones in a PoP-level ISP topology suggest that the proposed scheme improves the average

Manuscript received January 16, 2014.
Manuscript revised June 30, 2014.

[†]The authors are with School of Computer Science & Engineering, Southeast University, Nanjing 211189, China.
[††]The authors are with the Jiangsu Provincial Key Laboratory of Computer Network Technology, Nanjing 211189, China.
a) E-mail: xyhu@njnet.edu.cn
b) E-mail: jgong@njnet.edu.cn

DOI: 10.1587/transcom.E97.B.2360

*We use router, node and cache interchangeably as the entity with caching capability.

**We use data packet, content item and object interchangeably as the data entity for caching.

cache hit ratio of users' requests and reduces the average hop count as compared to CEE and the other on-path caching algorithms considered herein. Note that our proposed scheme can be applied to other ICN architectures with in-network caching as well.

2. Related Work

There is a large body of literature on collaborative caching [6]–[8] (off-path caching). Most consider an overlay model where collaborative caching is treated as an overlay service independent from the underlay networks, and they have limitations to be applied in NDN directly. These approaches either focus on special purpose applications which put additional constraints on the design (e.g., P2P system), or require the system to be constructed as a particular type of topology, e.g., a multicast tree. Extensive calculation is often required, which limits their usage in global environment. What is more, collaborative caching in NDN, if not well designed, could significantly increase the communication overhead as the number of in-network caches in NDN is considerable. Li et al. [11] treat the router-level topology of an ISP as a hierarchical tree and nodes in the tree make their caching decisions from top to down based on the caching decisions made by their parents and the content popularity information of the subtrees rooted at them, which asks the nodes in the tree to exchange information such as their content popularity and caching decisions. Recently Saino et al. [12] revisited hash routing techniques and proposed to determine the caching of an item by a random hash function on its content identifier. The scheme in [13] is similar but the router that caches a data chunk is decided by the chunk number modulo a predefined number. The two caching schemes without cache redundancy maximize cache hit ratio, but path stretch may incur a performance penalty as such caching considers nothing about the variance of users' requests at different routers.

On-path caching, which requires less coordination among caches, has attracted wide attention in the ICN research community and spawned interest in topics such as reducing caching redundancy [1], [2] and caching prioritization by popularity assessment [9], [10]. Chai et al. [1] proposed to choose the node with the highest centrality on the delivery path to cache a specific content item. Instead, Psaras et al. [2] suggested a probabilistic caching algorithm to provide fairness regarding the available capacity of the delivery path. The two on-path caching schemes do not prioritize content items and may result in unnecessary content replacement. In the caching schemes of both [9] and [10], an upstream router recommends the content to be cached at its downstream router based on the content popularity seen by the upstream router. We argue that each node itself is at a better position in prioritizing content among its users and this can reduce unnecessary redundancy, e.g., the replication of some content popular at certain neighbor but locally unpopular. Based on the idea that more popular content would serve a higher proportion of the total requests and caching content near users would reduce the average number of hops traversed by users' requests, this work proposes a distributed and opportunistic on-path caching. Each router independently decides whether to cache a data packet based on the local popularity of the data and the distances from the router or the content server to the consumer piggybacked in the data packet such that popular content is more likely to be cached by routers, especially routers near users, and cache redundancy is reduced.

3. Opportunistic On-Path Caching

3.1 The System Model

First let graph $G = (V, E)$ be the topology of an NDN domain where V and E are the sets of routers with caching capability and links in the domain separately, and let $O = \{o_1, o_2, \ldots, o_n\}$ represent the set of content requested by users and hosted by certain content servers outside the domain. 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, at least at the beginning of NDN, we restrict content caching to the en-route principle (i.e., users' requests are routed towards relevant content servers via shortest paths) as the first step towards a full-fledged one.

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. Namely, the requests for the same object 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 [14]. Hence, nodes in the network have distinct views of content popularity distribution. In our system, each router periodically obtains local content popularity distribution by statistics from users' data access history which reflects the requests aggregation and cache filtering effect.

A cache hit is recorded for a request if it finds a matching content along the content delivery path. Otherwise, a cache miss is recorded and the interest traverses the full delivery path to the content server. Without loss of generality, we assume that objects are of the same size and each cache slot in a content store can accommodate one object. In the event of the arrival of an uncached content, if the router determines to cache the newcomer and the content store is full, a cached object is replaced according to the replacement policy.

3.2 The On-Path Caching Algorithm

Under our opportunistic on-path caching, the probability of an en-route node caching a data packet is calculated based on the statistical popularity of the data observed by the node

Table 1 Model notation.

SYMBOL	MEANING
o_i	Solicited and returned data object
r_i	Local popularity of o_i observed by the router
β	The parameter configured by the operator
x	The count of hops traversed by the data packet for o_i so far
c	The count of hops traversed by the interest packet for o_i

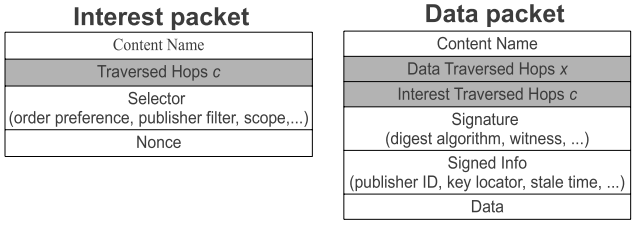


Fig. 1 Modified NDN packet types.

itself and the distances from the router or the content server of the data to the requester. The proposed opportunistic on-path caching at a router is sketched in Algorithm 1 and related symbols are explained in Table 1. β is a parameter configured by the operator of the network. The values of variables x and c are piggybacked in data packets. The value of the variable c is first stored in an added field of interest packets, the *Traversed Hops c* field, and each en-route router increases it by 1. Then it is copied into an added field of matching data packets, the *Interest Traversed Hops c* field, when the data packets are found. The value of the variable x is stored in another added field of data packets, *Data Traversed Hops x*, and each en-route router increases it by 1. Figure 1 illustrates the formats of the interest and data packets with the added fields.

Algorithm 1 Opportunistic On-path Caching (o_i)

Require: *Interest*(o_i) (request for content o_i)
  Update the statistic of o_i's popularity
  if o_i is in Cache **then**
    return_data (o_i)
  else
    if there is a matching PIT entry **then**
      Update the matching PIT entry
    else
      Increase the traversed hops c field in *Interest*(o_i) by 1
      Forward *Interest*(o_i)
    end if
    Get content object o_i back
    if have_enough_space (o_i) **then**
      add_to_cache(o_i)
    else
      Obtain r_i, x and c
      Compute $prob = r_i^{\beta} * \frac{x}{c}$
      Cache content with the probability $prob$
    end if
    Increase the value of x field of o_i by 1
    return_data (o_i)
  end if

Upon the arrival of *Interest*(o_i), an interest for o_i, the router updates its local popularity of o_i. If the interest gets a cache hit or a PIT entry match, data o_i is returned or the arrival face of the interest is added into the matching PIT entry. Otherwise, the interest gets a cache miss and is forwarded upstream. Then its traversed hops c is increased by 1 before being forwarded. When o_i returns (the router consults its content store and PIT to see if o_i is uncached and solicited), the router first checks if its cache space is not fully occupied yet. If yes, o_i is inserted into the cache directly. Otherwise, the router gets r_i from local popularity statistics and x and c piggybacked in data o_i. It computes $prob = r_i^{\beta} * \frac{x}{c}$ and caches o_i with the probability $prob$. Then the data traversed hops x of o_i is increased by 1 before o_i is forwarded downstream.

3.3 Analysis and Discussion

From Algorithm 1, we can see that as parameter β ($0 < \beta \leq 1$) increases, the same object is less likely to be cached at a specific router. We will evaluate the impact of parameter β on the performance of opportunistic on-path caching in Sect. 4. Here we claim that our opportunistic on-path caching possesses the following properties:

Proposition 1. Popular content is more likely to be cached by en-route routers.

PROOF. It can be seen that the probability of a router caching a data packet is proportional to the local popularity r_i and the data traversed hops x, and inversely proportional to the interest traversed hops c. Then for a popular object and an object with fairly small access frequency hosted by the same content server, a specific en-route router prefers the popular one as the popularity of the other is so small that its caching probability can even be negligible. Therefore, popular content is more likely to be cached and unnecessary replacement is reduced, which keeps popular objects in caches for longer and improves the cache hit ratio. □

Proposition 2. A specific data packet is more likely to be cached closer to its requester.

PROOF. For a specific data packet o_i, as it travels from its data source (either its content server or an intermediate cache) to the requester, its interest traversed hops c is constant, but its data traversed hops x increases. Moreover, due to the requests aggregation property, its popularity r_i may increase as it gets closer to its requester. Then as the probability of caching the data is proportional to r_i and x, the data is more likely to be cached in routers nearer to the requester, which reduces the average hop count of fetching the data. □

Proposition 3. An object that has been cached on the path from its content server to its requester is less likely to be cached by other routers on the path.

PROOF. Let the hops from the requester of the object to its content server S be c and the hops from the content server S to an en-route router R_a be x. If the object has not been cached on the delivery path, R_a caches the object with probability $prob = r_i^{\beta} * \frac{x}{c}$. If, instead, the object is cached by another router R_b on the path from S to R_a and R_b is α ($0 <$

$\alpha < x$) hops to S, then R_a caches the object with probability $prob^{'} = r_i^{\beta} * \frac{x-\alpha}{c-\alpha}$. Let us build a function $f(\alpha) = \frac{x-\alpha}{c-\alpha}$ where x and c are constants. Then its derivative $f^{'}(\alpha) = \frac{x-c}{(c-\alpha)^2}$. Since $x \leq c$, $f^{'}(\alpha) \leq 0$ and $f(\alpha)$ decreases as α increases. Thus $\frac{x}{c} \geq \frac{x-\alpha}{c-\alpha}$, viz., $prob = r_i^{\beta} * \frac{x}{c} \geq r_i^{\beta} * \frac{x-\alpha}{c-\alpha} = prob^{'}$. Hence, the object that has been cached on the path from the content server to the requester is less likely to be cached by another router on the path, which reduces cache redundancy. □

Proposition 4. For two objects with the same popularity in users, an en-route router prefers the one whose data source is farther from its requester.

PROOF. An en-route router caches an object o_i with probability $prob(r_i, x, c) = r_i^{\beta} * \frac{x}{c} = r_i^{\beta} * \frac{c-d}{c} = r_i^{\beta} * (1 - \frac{d}{c}) = PROB(r_i, d, c)$ where d is the hop count from the en-route router to the requester of o_i. Then for two objects with the same popularity $r^{'}$ at the en-route router and the same hop count $d^{'}$ from the en-route router to their requesters, the object with larger hop count from its requester to its data source, i.e., whose data source is farther away from its requester, is more likely to be cached by the en-route router. □

While only historical information is used, Famaey [15] showed the theoretical gain that can be achieved by prediction-based caching strategies under the assumption that perfect prediction about popularity within a certain period of time in the future is possible. With the ever changing access patterns at end users, routers have to periodically track content popularity. Fortunately, content popularity generally keeps stable within some time period and the popularity statistics can restart at regular time intervals. Generally speaking, a shorter interval is preferred by our caching scheme to quickly adapt the caching decisions to more dynamic access patterns. Note that our proposed caching algorithm is lightweight as each node independently makes its own caching decisions solely based on its own knowledge without requiring explicit information exchange with other nodes.

4. Performance Evaluation

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

4.1 Experiment Setup

We use the open-source ndnSIM [16] 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 2.70 GHz CPU with RAM 2.0 GB. We extend ndnSIM by adding *Traversed Hops* field in interest packets and *Data Traversed Hops* and *Interest Traversed Hops* fields in data packets , and by customizing the forwarding strategy in making caching decisions with our opportunistic on-path caching.

Network Topology. We run our simulations in two different network topologies — a small binary tree and a PoP-level ISP topology. A tree is small enough to be amenable to such analysis and instructive since from the view of an ISP, the content distribution topology is effectively a tree, and the simulations in the PoP-level topology reflect how the opportunistic on-path caching performs if deployed in real network.

Methodology. We evaluate how cache size, parameter β and users' access patterns impact the performance of the opportunistic on-path caching. In our simulations, we set homogeneous cache sizes at all routers as Rossi et al. [17] proved that the gain brought by content store size heterogeneity is limited. We measure the impact of parameter β by setting it from 0.3 to 1.0. We use synthetic request traffic whose popularity follows Zipf distribution with a typical exponent $s = 0.73$ (see, e.g., [5], [18]) as well as a real world Web traffic trace [19] as users' access patterns. Due to the stability of the access patterns considered herein, we treat the simulation duration as one interval of popularity statistics. We assume that users express interests as a Poisson process with constant average rates 100 interests/s. The payload size of data packets is 1024 bytes for all simulations and topologies. Each link in both topologies is assigned a bandwidth of 10 Mbps and a propagation delay of 1 ms. As the opportunistic on-path caching is independent of a specific cache replacement algorithm, similar to previous works [1]–[3], here we only present results with *Least Recently Used* (LRU). We also tried *Least Frequently Used* (LFU), which yielded qualitatively similar results.

Performance Metrics. The metrics to quantify the effectiveness of our algorithm are *cache hit ratio* (i.e., the percentage of interests satisfied by cached content) which implies upstream bandwidth demand, and *average hop count* (i.e., the average number of hops traversed by an interest packet) which demonstrates data delivery performance. We compare the proposed scheme (*Opportunistic*) against other on-path caching algorithms: (1) *CEE*, (2) a probabilistic algorithm which caches each content with probability 0.3 (*Prob(0.3)*), (3) a probabilistic algorithm which caches each content with probability 0.7 (*Prob(0.7)*), (4) the probabilistic algorithm (*ProbCache*) proposed in [2] and (5) the popularity based algorithm (*Popularity*) proposed in [10].

4.2 Small Scale Evaluations

We start our evaluation in a 3-level binary tree with 15 nodes as the simple router-level topology for an ISP. 8 content consumers are attached to the leaves of the tree and their interests are routed towards a single content server S attached to the root of the tree. There are 1000 distinct objects hosted by S and their popularity follows Zipf distribution with shape parameter 0.73 as shown in Fig. 2. Each simulation lasts for 8 minutes and is repeated for 10 runs to get the average results. Each run is set with different NS-3 "RngRun" arguments as seeds to randomize the request traffic and probabilistic caching.

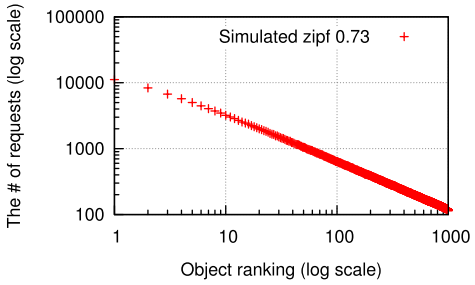
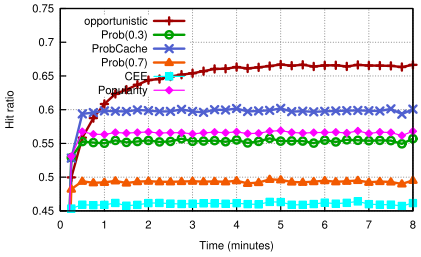
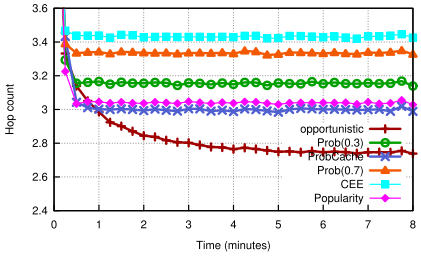


Fig. 2 The popularity distribution of the request trace for the binary tree.

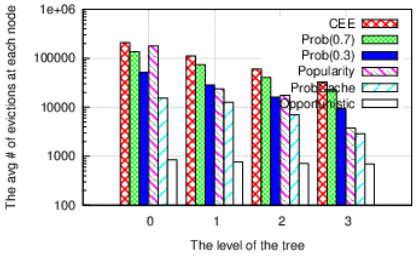


(a) Cache hit ratio.

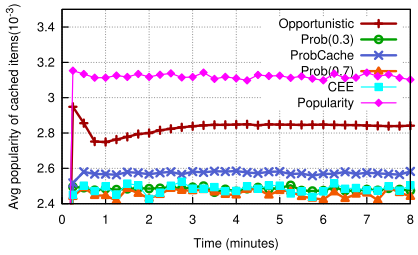


(b) Hop count.

Fig. 3 The average cache hit ratio and the average hop count in the binary tree. They are 0 and 5 separately for all algorithms at time 0 as there is no content in the caches.



(a) Evictions at each node of tree levels.



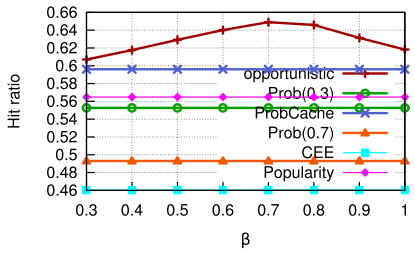
(b) Average global popularity of cached content.

Fig. 4 The average # of cache evictions at each node of different tree levels and the average global popularity of cached content in the binary tree.

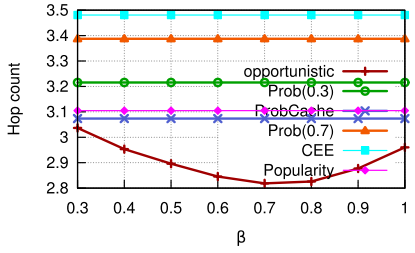
4.2.1 The Effectiveness of Opportunistic On-Path Caching

Figure 3 demonstrates the average cache hit ratio and the average hop count of users' requests in the binary tree under different caching algorithms when the cache sizes of the routers are set to 100 objects and β is set to 0.7. It can be seen that it takes our proposed caching scheme less than one minute to initially learn the data access patterns of these users and then it outperforms the other caching algorithms considered herein in improving the average cache hit ratio and reducing the average hop count. In particular, it improves the average cache hit ratio from 0.46 to 0.67 (an improvement of approximately 0.21) and reduces the average hop count from 3.4 to 2.7 (a reduction of 0.7) against CEE; there is around 0.08 $\sim$ 0.17 improvement in the average cache hit ratio and around 0.3 $\sim$ 0.6 reduction in the average hop count compared to the other algorithms. As the content server is attached to the root node of the tree and near users in the scenarios evaluated herein, the reduction in the average hop count is unobvious. But in real scenarios where content servers may be farther from users, the reduction in the average hop count should be more positive.

To understand how the opportunistic on-path caching actually works, we measure the average number of cache evictions at each node per tree-level (where the smaller the tree level, the closer to the content server this level is) and the average global popularity (i.e., the content popularity among all users) of cached content. As shown in Fig. 4(a), our proposed scheme reduces the number of cache evictions from hundreds of thousands under CEE to less than a thousand, a reduction of about 2 orders of magnitude. The big reduction in the number of cache evictions validates our claim that the opportunistic on-path caching algorithm reduces cache space contention. As seen in Fig. 4(b), after the system reaches steady state, the average global popularity of cached content under the opportunistic on-path caching is larger than that of the rest algorithms except the popularity based caching by more than 3×10^{-4}. Furthermore, there are another two surprising observations. The first is that under our caching scheme, the average global popularity of cached content in the first 30 seconds is larger than that in the time remaining. The reason is that at the beginning, these caches are empty and the accessed content items which should be relatively popular are duplicately stored in caches on their delivery paths; then later as more objects are accessed by users, our scheme picks other content to store in caches to increase the diversity of cached content, which improves the average cache hit ratio. The second observation is that the popularity based caching beats our proposed caching scheme in terms of the average global popularity of cached content throughout the simulation duration, whereas its average cache hit ratio is smaller. Then we further measure the cache redundancy of these algorithms (to save space, the related figure is not shown) and find that there is much

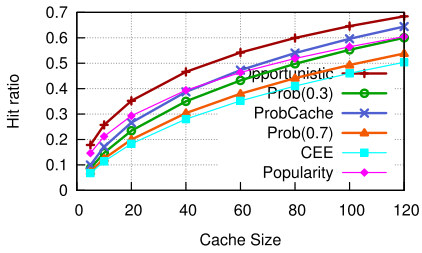


(a) Cache hit ratio.

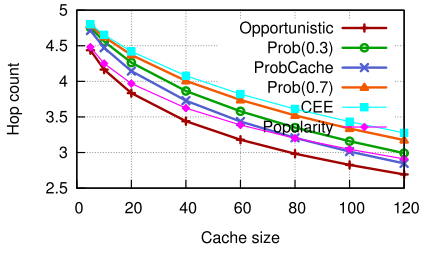


(b) Hop count.

Fig. 5 The impact of parameter β on the average cache hit ratio and the average hop count in the binary tree.



(a) Cache hit ratio.



(b) Hop count.

Fig. 6 The impact of cache size on the average cache hit ratio and the average hop count in the binary tree.

more cache redundancy under the popularity based caching, which may be unnecessary, e.g., a node is asked to cache certain content popular at a neighbor but locally unpopular. Therefore, our caching scheme not only succeeds in picking relatively popular content to stay in caches, but also reduces unnecessary cache redundancy.

4.2.2 The Impact of Parameter β and Cache Size

Figure 5 plots the impact of parameter β on the average cache hit ratio and the average hop count for simulations within 8 minutes where the cache sizes of the routers are set to 100 objects. It can be seen that with all evaluated setting of β, our opportunistic on-path caching performs the best among all the caching algorithms considered herein and when β is set to 0.7, its improvement in the average cache hit ratio is up to 0.19 and its reduction in the average hop count is up to 0.7 against CEE. But if β is set to 0.3, there is no significant difference in the caching performance between our scheme and ProbCache. Hence, if the opportunistic on-path caching is deployed in real network, operators should adapt the β value to users' access patterns.

To evaluate the impact of cache size on the performance of the proposed caching scheme, we set β to 0.7 and change the cache sizes from 5 to 120 objects. Figure 6 illustrates the results of the average cache hit ratio and the average hop count versus cache size. As expected, with different settings of cache size, our scheme still outperforms the others. Its improvement in the average cache hit ratio is around $0.11 \sim 0.19$ against CEE and around $0.03 \sim 0.16$ compared to the other algorithms. Its reduction in the average hop count is roughly $0.36 \sim 0.64$ against CEE and around $0.04 \sim 0.57$ against the rest of the algorithms. Furthermore, as the cache size increases, the performance of the proposed scheme is improved, but at a slower pace since the objects inserted into the additional cache space is less popular due to the Zipf distribution nature of users' request popularity.

4.3 Larger Scale Simulations

In this subsection, we investigate the behavior of our opportunistic on-path caching in a more practical and larger scale network topology using a real trace. The simulations here model the scenario where there are multiple content servers each hosting a different content set, users access content from these content servers and content items are opportunistically cached in network during users' access. The adopted topology is a PoP-level Rocketfuel SPRINT topology [20] with 52 nodes as displayed in Fig. 7 where nodes with degree 1 are classified as consumers (19 red nodes), the nodes directly connected to the consumers are classified as gateways (13 green nodes), and the remaining nodes are classified as intermediate routers (20 white nodes). 3 outside content servers each are attached to a different gateway randomly picked for each simulation run (we conduct 10 simulation runs for each simulation). Since we do not find a trace which includes users' access of different content sets, to remedy this, we adopt a WorldCup request trace [19] used in [21] and divide the set of requested objects in the trace into three subsets each serving as the content set hosted by one of the three content servers. The trace includes $2,880,720$ requests and $7,175$ distinct objects and its object popularity distribution is shown in Fig. 8.

We evaluate the impact of parameter β on the performance of our proposed scheme and find that our caching scheme with $\beta = 0.7$ still works the best in this specific scenario (due to space limitation, we do not include the re-

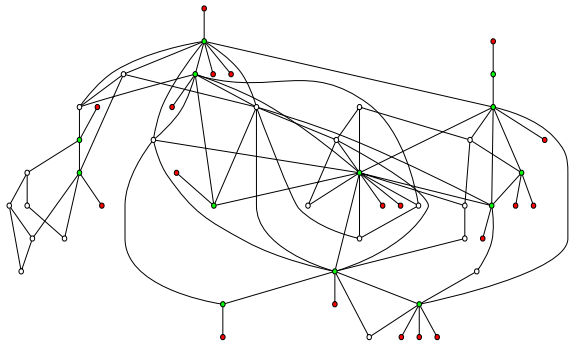


Fig. 7 The PoP-level SPRINT topology.

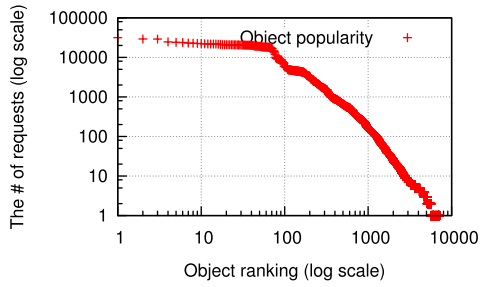
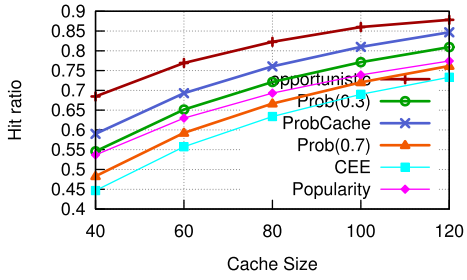
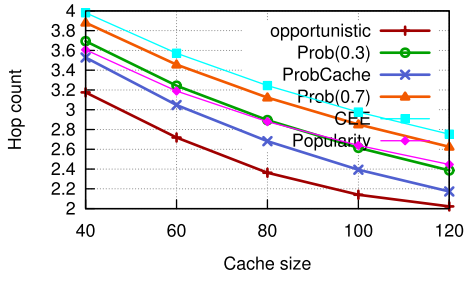


Fig. 8 The popularity distribution of the WorldCup request trace.



(a) Cache hit ratio.



(b) Hop count.

Fig. 9 The simulation results under different caching algorithms in the SPRINT topology.

lated figure). To evaluate the impact of cache size, we keep setting cache size from 40 objects to 120 objects and β to 0.7. As illustrated in Fig. 9, in this specific scenario where there are multiple content servers with different distances to the users, the opportunistic on-path caching still stands out. Its improvement in the average cache hit ratio is up to 0.24 against CEE and around 0.04 ~ 0.20 compared to the other algorithms. Its reduction in the average hop count is roughly 0.8 against CEE and around 0.3 ~ 0.8 against the others.

5. Discussion & Future Work

Our opportunistic on-path caching, a lightweight scheme that does not require explicit information exchange among caches, has been verified to be effective in improving the average cache hit ratio and reducing the average hop count traversed by interests. It is now, however, vital to detail some key limitations of the study. The first limitation is the lack of realistic routing information for NDN. NDN routing is usually based on a range of characteristics including load balancing and QoS issues. However, as NDN has not been deployed, there is no information regarding this available and so far, the routing in ndnSIM is still based on fixed shortest paths, thereby removing the presence of unpredictable multipath routing and route variations. The second limitation is that we assume the content items being studied are with equal size, which may not be that practical, whereas the distribution of object sizes may make some difference in in-network caching. The third limitation is the memory consumption for popularity statistics which is proportional to N, the number of distinct objects accessed 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, we can use space efficient double counting Bloom filter [22] for recording content popularity to further reduce space usage, which is similar as that in [22] for making a statistic of flow length on high speed network. The above mentioned point out the direction of refining our on-path caching algorithm and further verifying it.

6. Conclusion

This work proposes a distributed and opportunistic on-path caching scheme for Named Data Networking domains to reduce upstream bandwidth demand and improve data delivery performance. It asks each en-route router to independently decide the probability of caching a specific content item based on the data popularity observed by the router itself and the distances from the router or the content server to the requester such that popular content is more likely to be cached by routers, especially routers near users. We verify the effectiveness of our proposed scheme by conducting simulations in both a binary tree and a PoP-level ISP topology. We use both synthetic request traffic whose popularity follows Zipf distribution and real world Web traffic trace as users' access patterns of our simulations. Our extensive simulations suggest that the proposed scheme improves the average cache hit ratio and reduces the average hop count traversed by interests as compared to the other on-path caching algorithms considered herein. Moreover, we examine the actual cache responses and verify that the proposed scheme does reduce unnecessary cache replacement and cache redundancy, and pick popular content to keep in caches. Our

next step is to refine our on-path caching by taking into account multipath routing and object size variation.

Acknowledgements

This work was 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 Study on Future Network 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] W.K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," Proc. IFIP'12, pp.27–40, Berlin, Heidelberg, 2012.
[2] I. Psaras, W.K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," Proc. ICN'12, pp.55–60, New York, NY, USA, 2012.
[3] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard, "Networking named content," Proc. CoNEXT'09, pp.1–12, New York, NY, USA, 2009.
[4] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," IEEE Commun. Mag., vol.50, no.7, pp.26–36, 2012.
[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," Proc. INFOCOM'99, pp.126–134, New York, NY, USA, 1999.
[6] J. Ni and D.H.K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," IEEE Commun. Mag., vol.43, pp.98–105, May 2005.
[7] P. Sarkar and J.H. Hartman, "Hint-based cooperative caching," ACM Trans. Comput. Syst., vol.18, no.4, pp.387–419, 2000.
[8] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," Proc. INFOCOM'01, pp.1416–1424, Anchorage, Alaska, USA, 2001.
[9] 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," Proc. NOMEN'12, pp.316–321, Orlando, Florida, USA, March 2012.
[10] C. Bernardini, T. Silverston, and F. Olivier, "Towards popularity-based caching in content centric networks," RESCOM 2012, pp.1–2, Les Vosges, France, June 2012.
[11] 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," Proc. ANCS'12, pp.15–26, Austin, TX, USA, 2012.
[12] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," Proc. ICN'13, pp.1–6, Hong Kong, China, 2013.
[13] Z. Li and G. Simon, "Time-shifted tv in content centric networks: The case for cooperative in-network caching," Proc. ICC2011, pp.1–6, Kyoto, Japan, 2011.
[14] C.L. Williamson, "On filter effects in web caching hierarchies," ACM Trans. Internet Techn., vol.2, no.1, pp.47–77, 2002.
[15] J. Famaey, T. Wauters, and F.D. Turck, "On the merits of popularity prediction in multimedia content caching," Proc. IM'11, pp.17–24, Dublin, Ireland, 2011.
[16] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," Technical Report NDN-0005, NDN, Oct. 2012.
[17] D. Rossi and G. Rossini, "On sizing ccn content stores by exploiting topological information," Proc. NOMEN'12, pp.280–285, Orlando, Florida, USA, March 2012.
[18] H. Gomaa, G. Messier, R. Davies, and C. Williamson, "Media caching support for mobile transit clients," Proc. WIMOB'09, pp.79–84, Washington, DC, USA, 2009.
[19] "Ircache home," http://ita.ee.lbl.gov/html/traces.html
[20] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," ACM SIGCOMM Comput. Commun. Rev., vol.32, no.4, pp.133–145, 2002.
[21] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," Proc. INFOCOM'12, pp.2426–2434, Orlando, Florida, US, 2012.
[22] H. Wu, J. Gong, and W. Yang, "Algorithm based on double counter bloom filter for large flows identification," J. Software, vol.21, no.5, pp.1115–1126, 2010.

Xiaoyan Hu got her B.S. degree in software engineering from Nanjing University of Science & Technology in 2007 and her M.S. degree in computer architecture from Southeast University in 2009. She is now a Ph.D. candidate in Southeast University focusing on the design of NDN in-network caching. She visited netsec lab in Colorado State University, a research group working on NDN, from Sept. 2010 to Aug. 2012.

Jian Gong is a professor in School of Computer Science and Engineering, Southeast University. His research interests are network architecture, network intrusion detection, and network management. He received his B.S. in computer software from Nanjing University, and his Ph.D. in computer science and technology from Southeast University.