PPP: prefix-based popularity prediction for effective caching in content-centric networking

Article · June 2014

DOI: 10.1145/2619287.2619296

7 authors, including:

Bing Han

Seoul National University

2 PUBLICATIONS **12** CITATIONS

SEE PROFILE

Xiaofei Wang

Seoul National University

47 PUBLICATIONS **465** CITATIONS

SEE PROFILE

Ted "Taekyoung" Kwon

Seoul National University

55 PUBLICATIONS **680** CITATIONS

SEE PROFILE

PPP: Prefix-based Popularity Prediction for Effective Caching in Content-Centric Networking

Bing Han[†],Xiaofei Wang[♠],Xin Chen[†], Ted "Taekyoung" Kwon[†],Yanghee Choi[†], Dung Ong Mau[°], Min Chen[°]
[†]School of Computer Science & Engineering, Seoul National University, Korea.
[♠]Department of Electrical and Computer Engineering, The University of British Columbia, Canada.
[°]School of Computing Science and Technology, Huazhong University of Science and Technology, China.

ABSTRACT

In the Content-Centric Networking (CCN) architecture, popular content can be cached in some intermediate network devices while being delivered, and the following requests for the cached content can be efficiently handled by the caches. Thus, how to design in-network caching is important for reducing both the traffic load and the delivery delay. In this paper, we propose a caching framework of Prefix-based Popularity Prediction (PPP) for efficient caching in CCN. PPP assigns a lifetime (in a cache) to the prefix of a name (of each cached object) based on its access history (or popularity), which is represented as a Prefix-Tree (PT). We demonstrate PPP's predictability of content popularity in CCN by both traces and simulations. The evaluation results show that PPP can achieve higher cache hits and less traffic load than traditional caching algorithms (i.e., LRU and LFU). Also, its performance gain increases with users of high mobility.

Keywords

Content-Centric Networking, Popularity-Prediction, CCN Caching Simulator

1. INTRODUCTION

The Internet traffic has been explosively growing with the increasing demand for massive content distribution and large amounts of media resources. The growth of the traffic load poses a significant burden and thus challenges the current Internet infrastructure while users often suffer from severe network congestions and delays. To solve this problem, Content-Centric Networking (CCN) [1], has been proposed and gained much attention due to its attractive advantages, such as (1) content is located by name instead of by location, and (2) while forwarding contents, every CCN node can cache the requested content.

In CCN, the name of a (content) object is of a hierarchical structure and human readable [2], and a content delivery is realized by the request of the corresponding object. For this, each CCN node maintains three data structures: *Forwarding Information Base* (FIB), *Pending Interest Table* (PIT) and *Content Store* (CS). The details of the FIB, PIT and CS can be referred to [3]. The name-based forwarding and routing make CCN quite different from the current TCP/IP architecture. Thus, designing new caching strategies becomes crucial in the research community, and this new challenge should take into account the characteristics of content popularity, the users, and the CCN nature.

The prior studies in this area mostly focus on the caching decision algorithm, and the caching replacement algorithm , such as *Least Recently Used* (LRU) and *Least Frequently Used* (LFU); the latter of which are originated from the computer architecture literature. The rationale behind LRU and LFU is that most of popular objects are consumed many times once they are used, and their popularity generally follows Zipf distribution, which reflects the *power-law* effect [4], That is, a relatively small number of popular objects are requested by a large number of users, which account for the majority of the Internet traffic.

Thus, our conjecture is that, if the in-network caching can detect and cache popular objects more timely (i.e. faster that other schemes), the dissemination of popular objects will be much more efficient. We also note that the names of popular objects may often belong to and start with the same domain name (and often the same overlapping pathname as well); for instance, `cnn.com/news/asia/news1.jpg` and `cnn.com/news/asia/news1.avi` can be popular simultaneously. Every possible string from the beginning character (of a name) to a character right before any slash can be a prefix: e.g. `cnn.com/news` and `cnn.com/news/asia` in the above examples. Hence, content names may have various lengths and levels of prefixes, which is detailed in [5]. Note that LRU and LFU do not consider the relation among the name prefixes of the popular objects, and thus they cannot recognize the popular prefix of a newly generated object until it will be sufficiently requested over some duration. It is well-known that popular publishers tend to make popular objects, and they can also explosively increase the popularity of unpopular objects (from not-so-popular publishers) merely by publishing them under their names [6], [7]. Therefore, we are motivated to propose more efficient caching policy that can consider the above practice of popular objects and popular publishers. That is, we seek to cache objects faster that could be popular with the higher popularity.

In this paper, we propose a Prefix-based Popularity Prediction (PPP) algorithm for CCN. The central idea of PPP is that if popular objects share the same prefix, that prefix will be even more popular due to aggregation. For instance, suppose that multiple objects of the same domain name are popular. Then we guess that a new object with the same domain name can be popular with a high probability. Considering the hierarchical tree structure of *names and their prefixes* of content objects, we build a tree structure, called a *Prefix Tree* (PT) to handle prefixes and their lifetimes. At a CCN

CFI'14, June 18 - 20 2014, Tokyo, Japan

http://dx.doi.org/10.1145/2619287.2619296 .

node, PPP can maintain a PT for popular objects; the PT is used to analyze the popularity of not only the individual objects but also the hierarchical groups of objects (who share the same prefix). PPP then allocates a suitable lifetime for each prefix that may include not-yet-generated objects. PPP operates compatibly with in the current CCN architecture and adds no functional requirements. We develop a new tool named *CCN Caching Simulator* to test and improve the caching policy. We test two trace datasets with this tool, which shows that PPP outperforms the traditional caching policies (e.g. LRU and LFU).

The rest of this paper is organized as follows: Section II briefly introduces some previous studies on caching in CCN; In Section III, we describe details of the Prefix-based Popularity Prediction (PPP) algorithm; We verify the PPP framework by traces in Section IV; The paper is concluded in Section V.

2. RELATED WORK

It is estimated that trillions of contents exist on the Internet nowadays, and it is known that the probability distribution of requests for different content objects is very skewed [4] [8]. Z. Ming et. al., [10] show that top 5% of videos contribute over 80% of views in Youku (which is the Youtube of China). The implication is that the caching can be very efficient by storing only 5% to 10% of the popular videos.

Then there have been some studies focusing on the popularity-based caching policies and performance in CCNs. Z. Ming et al., [10] present the aged-based cooperative caching policy, by which the replica is stored near the network edge; the more popular a replica is, the longer (caching) lifetime it has. The study in [8] also discusses the similar popularity-based dynamic cache management. W. Zhang et al., [5] consider the popularity of IP prefixes at border routers with the power law distribution. J. Li et al., [8] consider the popularity skewness over a range from 0.5 to 2. Also Z. Li et al., [11] propose to use collaboration between caches, and the LRU/LFU policy is tested in this paper. Kideok et al. [12] proposed a caching scheme named WAVE, in which the popular contents can be pushed closer to the end users. However, WAVE cannot eliminate the caching redundancy and is limited to achieve good caching performance with prediction.

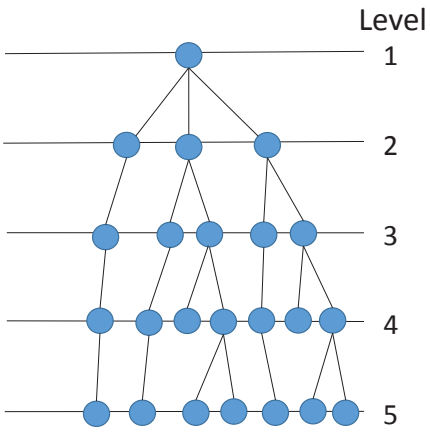
Furthermore, regarding the object popularity, the caching at the network edges is more essential and important. A. Ghodsi et al., [13] have suggested that most of the benefit of CCN is achieved at edge routers/gateways and fast decreases as caching places become distant from edges. G. Tyson et al., [14] have tested the simulation to show the effectiveness of a simple edge cache of 10,000 packets. It would allow 11% of *Interests* to serve the requests from the same Autonomous System (AS).

3. PPP DESIGN

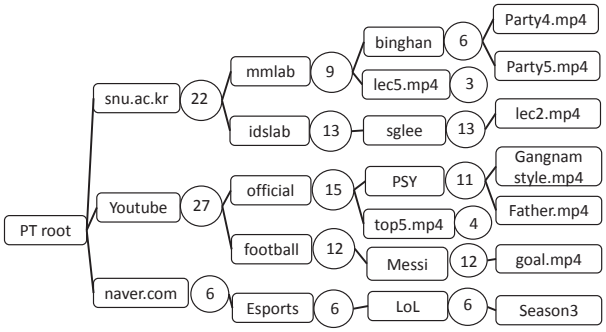
In the current TCP/IP networking, an ISP caches and shares content objects to improve the utilization of their facilities. In CCN, routers need to consider more economical incentives and scenarios for caching and sharing content objects efficiently. In this section, we present the details of our PPP algorithm design.

3.1 Prefix Name

Fig. 1(a) illustrates the structure of prefix names with 5 levels. The root on the first level is not associated with any particular name or prefix; it is similar to the root server in the DNS. The second level corresponds to the domain names. The administrator of a domain (or the publisher) provides and broadcasts her domain name across the whole network like BGP. A CCN node adds the domain name



(a) Name tree structure with 5 levels of prefixes



(b) A example of PT's structure

Figure 1: Prefix Tree's structure is illustrated

into its routing table (FIB), and forwards it to other nodes. All the CCN nodes can update their FIBs in case of the network changes (like link up/down in BGP). The third level of the tree refers the the top level directory of the path name of an object; for instance, `news` in `cnn.com/news/asia/logo.jpg`. Thus, there can be multiple levels in content names; however, we illustrate up to 5 levels here.

3.2 Prefix Tree Architecture

To keep track of the popularity of prefixes in content names, we further construct one more data structure, called *prefix tree* (PT). Thus the PT is to maintain the prefixes, their popularity potentials, and their lifetimes. That is, if some objects are recognized as popular ones, CCN nodes will keep them alive in the CS longer than the others, in order to achieve a higher cache hit ratio. PPP finds popular objects by counting the matched prefixes for each *Interest* being forwarded. Fig. 1(b) shows an instance of the PT's structure. For instance, once there is an incoming *Interest* whose content name is "*ccnx://www.snu.ac.kr/talk/van/ccn.pdf*", the CS will increase the counters of the matching prefixes as follows:
$C_p[2]$("*www.snu.ac.kr*")++
$C_p[3]$("*www.snu.ac.kr/talk*")++
$C_p[4]$("*www.snu.ac.kr/talk/van*")++
$C_p[5]$("*www.snu.ac.kr/talk/van/ccn.pdf*")++

where $C_p[i]$ is the counter for the i^{th} prefix level. Then, the life-

time of each content object is calculated by the following equation:

$$t_l = t_u \times \sum_{i=2}^{5} \left(w[i] \times C_p[i] \right) \quad (1)$$

where t_u is a lifetime unit variable detailed below, and $w[i]$ is the weight of $C_p[i]$ determined empirically. (We found that PPP performs well by setting $w[i]$ to be proportional to i.) Our conjecture is that high cache hit ratio will be achieved as popular objects are located in the CS longer. At the *CS* refreshing time, if the *CS* still has free space to store more content objects, it will not delete any content object even if the lifetime of the object is already over.

The rate of arrival of *Interests* (f_{req}) is an important metric that strongly effects the replacement algorithm. But it may be fluctuating as the number of users varies over time in practical scenarios. We adapt to dynamic request rates from users by controlling a lifetime unit t_u. The t_u value is adaptively adjusted every period (say, 60 seconds) based on measuring the number of arriving *Interests*. When f_{req} value becomes higher, the *CS* will be replaced faster. To avoid LRU/LFU deleting popular objects, we decrease t_u as f_{req} increases. Decreasing t_u helps to reduce the lifetimes of all the cached objects. Then, unpopular objects can be found faster and be replaced before new objects arrives. When f_{req} value is low, we increase t_u value to keep objects in the *CS* as long as possible. Therefore, $t_u \times f_{req}$ can be a constant.

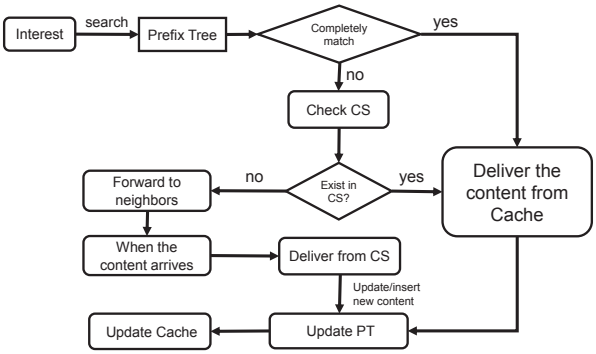


Figure 2: The flowchart of the PPP algorithm is shown

3.3 PPP Algorithm

Upon receiving an *Interest*, the CCN node applies the PPP algorithm. The process is illustrated in Fig. 2, and its steps are merged into the CCN forwarding algorithm. If the *Interest* makes a completely match with a PT entry, it means that the *requested content* (RC) is cached in the *CS* and can be directly delivered; otherwise, we should check whether the RC exists in the *CS*, because the RC might be unpopular but still stored in the *CS*. The *Interest* will be forwarded to the next router if the RC is not in the current router. When the RC is fetched from other nodes, it will be delivered back and the RC's information will be inserted to the PT by automatically updating relevant prefix entries. As time goes on, popular objects will keep attracting a large amount of requests, and thus their lifetimes will be longer. Also, even a newly generated object that has a common popular prefix with the legacy popular objects can be quickly cached, which helps to achieve more efficient caching.

4. TRACE PERFORMANCE EVALUATION

To evaluate the effectiveness of PPP, we select two trace datasets, BitTorrent trace (B-trace) and IRCache trace (I-trace), which are publicly available trace datasets that contains both user activity records and prefix of content objects. We implemented a new tool named *CCN Caching Simulator* (CCS), which evaluates and validates the performance of PPP, to compare it with LRU/LFU.

4.1 CCN Caching Simulator

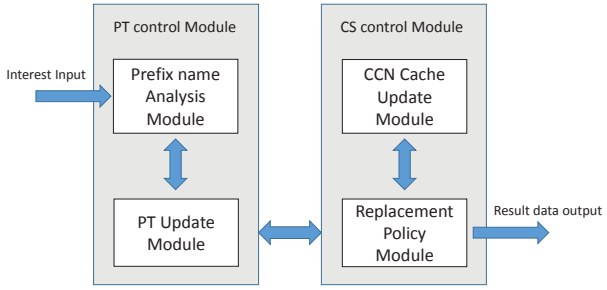


Figure 3: CCN Caching Simulator architecture

We implemented CCS with C programming on ubuntu (version 12.04 LTS) following the specifications in the previous sections. B-trace and I-trace datasets are embedded in CCS module. There are two components in CCS, PT control module and CS control module. We discuss the following submodules in CCS structure design:

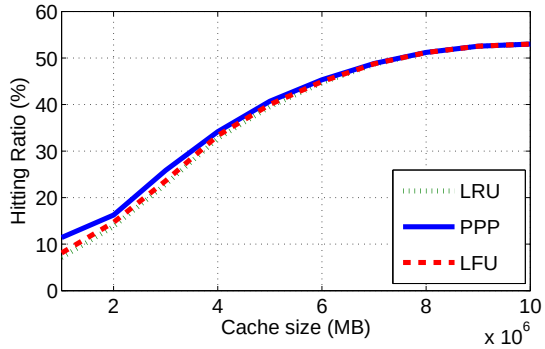
- **Prifix name Analysis Module**: The new coming request should be separated by prefix names. In each prefix level, the prefix name is recognized as popular when it is matched in PT.
- **PT Update Module**: While the prefix of coming request is recognized as popular, it will be updated into PT and all the content objects related to this prefix will be updated in order to maintain the lifetime.
- **Replacement Policy Module**: The lifetime of new content objects will be compared with other existing objects by caching policy settings. The policy can be modified or replaced conveniently and there is no influence on other modules.
- **CCN Cache Update Module**: Based on the replacement policy module, the content objects in CCN cache can be automatically updated.

PT control Module and CCN Cache Update Module have been implemented as the new API function. In that case, the data trace can be directly used by API function such like *getcategory()* from lib *Category.c*. To significantly benefit users, we set the PPP, LRU and LFU methods as the default cache policy and directly use them by API in CCS so that users can compare it with the new cache policy, which is designed into *Replacement Policy Module* by themselves, to obtain the different performance.

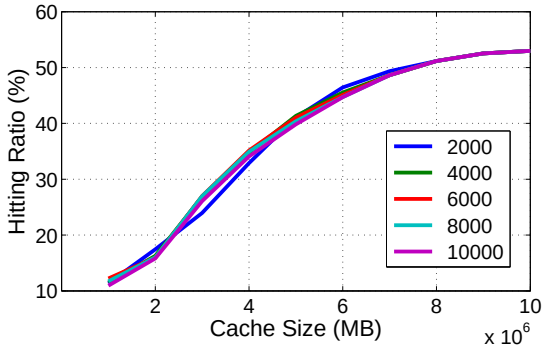
There are three benefits of CCS: (1) Convenience, the new cache policy can be replaced easily in CCS; (2) Compatibility, varied trace datasets can be used in CCS; (3) Scalability, functions as required can be improved in CCS.

4.2 BitTorrent Trace

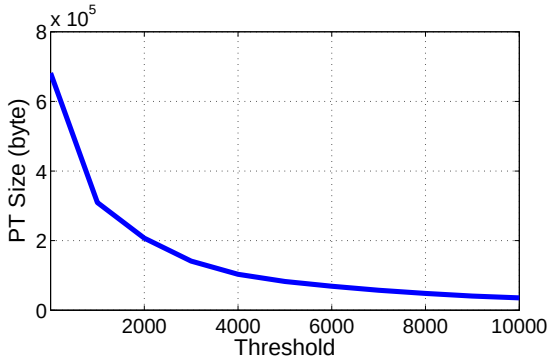
In this experiment, our simulation is based on using B-trace which is obtained from real server measurement in [15].



(a) The CS policies comparison with varied cache sizes



(b) The multiple threshold policies of PT with varied cache sizes

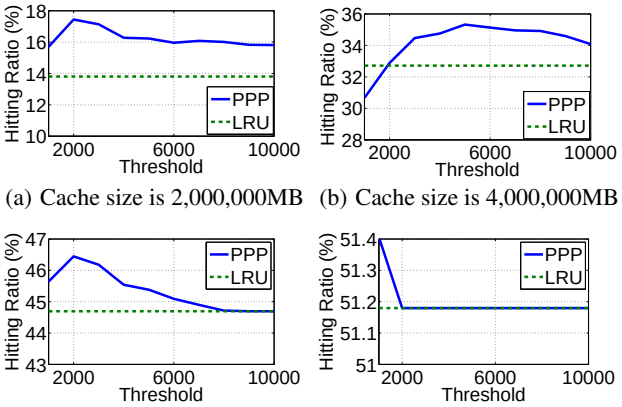


(c) The state of PT size with varied threshold when cache size is 2,000,000MB

Figure 4: The relationship of Threshold, PT size and Hitting Ratio for Bit-Torrent trace

The B-trace dataset is collected for 77 days from Feb.14th to May.1st, 2011. The information of the B-trace consists of user IP, folder's name in each prefix level, request time and file size. Based on the B-trace, we study 2 kinds of replacement caching policies, which are LRU and PPP. We use 100,000 content objects with over 8 million requests, the total size of content objects is over 120 TB. The cache size of a CCN node is set from 1 TB to 10 TB. A metric parameter named *threshold* is used to restrict and adjust the total number of content objects in the cache, contributing to obtain the impact of cache size on the performance for various replacement caching policies. The value of threshold is set from 1000 to 10000.

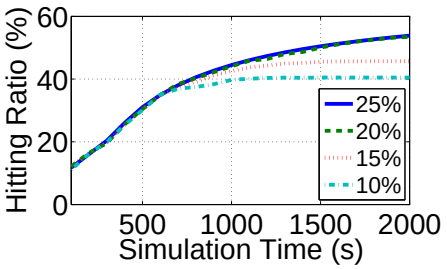
We measure the hitting ratio with varied cache size, as shown in Fig. 4(a) where PPP outperforms LRU/LFU with the higher cache cache hit ratio when the cache size is smaller than $6 * 10^6$MB. (Since the LFU has similar hit ratio with LRU in some cases, it is not shown in some figures.) With the increase of the cache size, the performance gap between LRU and PPP is reduced. With such a trend, it's easy to understand that all of the replacement caching policies seldom perform content replacement when the cache size is large, causing similar performance in terms of cache hit ratio. As Fig. 4(b) shows, the curve of cache hit ratios with varied threshold are similar. The PT size is related to the threshold's value. The Fig. 4(c) clearly shows that PT size is reduced while threshold increases, because the most of content objects are stored in cache, and thus decreasing the advantage of building the PT while introducing tree-construction overhead.



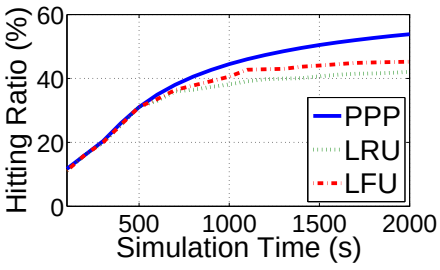
(a) Cache size is 2,000,000MB (b) Cache size is 4,000,000MB

(c) Cache size is 6,000,000MB (d) Cache size is 8,000,000MB

Figure 5: The comparison of Bit-Torrent Trace between PPP and LRU with multiple relative cache sizes



(a) PPP with multiple relative cache sizes

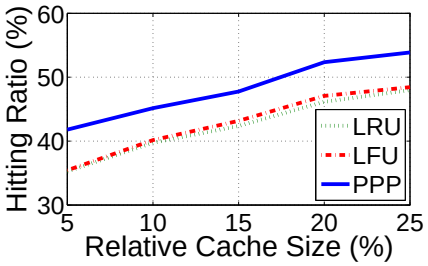


(b) Multiple cache policies with threshold value 3000

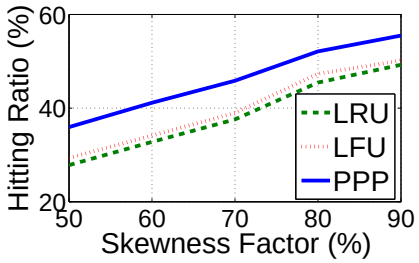
Figure 6: PPP with multiple relative cache sizes

In Fig. 5, we fix cache size to different values while varying threshold. The Fig. 5(a) shows that when the cache size is 2,000,000MB, PPP can gain 16% higher cache hit ratio than LRU.

In Fig. 5(b), while the cache size increases to 4,000,000MB, PPP also obtains a higher cache hit ratio than LRU but the difference value decreases. While the cache size increases to 6,000,000MB, the Fig. 5(c) shows that PPP loses its effectiveness while the cache size is large, As shown in Fig. 5(d), PPP has no effectiveness for cache utilization while the cache size is too large.

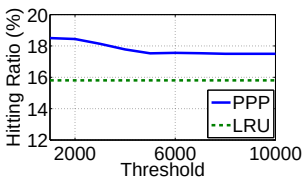


(a) Final state of cache policies with varied relative cache sizes

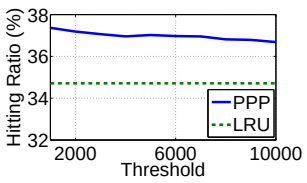


(b) Impact of popularity skewness on hitting ratio with relative cache size 20%

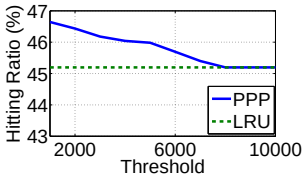
Figure 7: Final state of cache policies and impact of popularity skewness on cache hit ratio



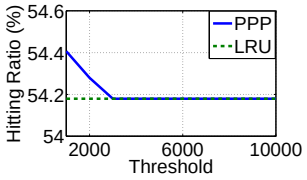
(a) Cache size is 2,000,000MB



(b) Cache size is 4,000,000MB



(c) Cache size is 6,000,000MB



(d) Cache size is 8,000,000MB

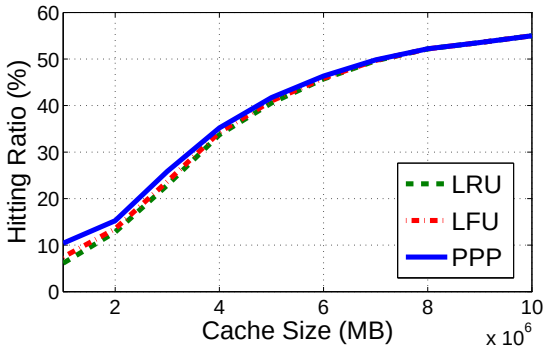
Figure 8: The comparison of IRCache trace between PPP and LRU with multiple relative cache sizes

For the impact of the cache size on cache hit ratio, we compare the effectiveness of PPP with varied relative cache sizes. Fig.6(a) shows that the cache size at a CCN node is set with 10%, 15%, 20% and 25%, the values of cache hit ratio at the final state are 38%, 44%, 52% and 53%, respectively. The results show that a higher cache hit ratio can be obtained when we expand a cache volume, and the growing of cache hit ratio does not linearly increase with cache volume. Fig.6(b) illustrates the effect of two replacement policies with 20% relative cache size. The results show that PPP produces the higher cache hit ratio when simulation time increases and the simulation approaches to a stable state.

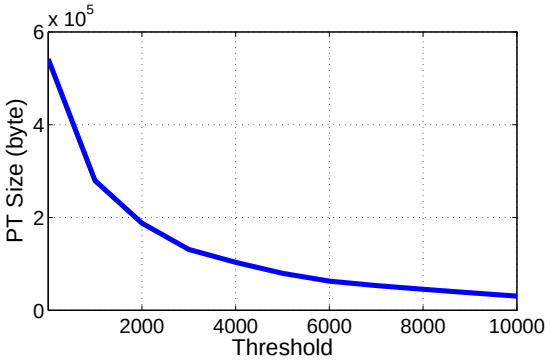
We analyze the impact of replacement policy on cache hit ratio with various relative cache size. In Fig.7(a), when the relative cache size increases to 10%, the difference of the maximum value of the hitting ratio between PPP and LFU is up to 10%. But the different value will reduce to 8% and 6% while the relative cache size increases to 20% and 25%. That is, PPP yields higher performance than LRU with varied cache size, but performance increment reduces with the decrease of the relative cache size. Fig.7(b) shows the performance of two replacement policies with the 20% relative cache size. While the value of skewness is 80%, the curve of cache hit ratio pitches sharply and the value of cache hit ratio obviously increases. That is, PPP always gains much better performance than traditional algorithms and the cache hit ratio is largely concentrated in the most popular objects while the skewness factor increases.

4.3 IRCache Trace

We also carry out the same experiments with I-trace [16]. The period of I-trace collection is from October 15th to October 17th, 2013. We use more than 70000 content objects with over 500000 requests. The total size of content objects is about 70TB. The other settings are the same as in the experiments of B-trace.



(a) The CS policies comparison with varied cache sizes



(b) The state of PT size with varied threshold when cache size is 2,000,000MB

Figure 9: The relationship of Threshold, PT size and cache hit Ratio for IRCache trace

Fig. 8 shows comparison of the performance metrics between PPP and LRU with varied cache size. In general, PPP outperforms LRU with higher cache hit ratio when cache size is small. Howev-

er, the different value between PPP and LRU decreases due to the cache size increasing because the most of content objects will be stored in the cache so that the popularity prediction will become less effective. Fig. 9(a) also shows that PPP outperforms LRU when the cache size is small. In Fig. 9(b), the PT size decreases with a larger threshold. The other performance analysis is similar with the case of B-trace.

5. CONCLUSION

In this paper, we proposed a new approach for caching decision and replacement that can be applied for CCN. The proposed algorithm PPP allows CCN nodes to keep track of the prefixes of popular objects, so that CCN nodes can achieve higher efficient caching. The simulation results show the great benefits of PPP in terms of the cache hit ratio, low cost and reduced latency compared to traditional LRU/LFU based policy. As the future work, we are planning to investigate the PPP performance in various settings: (1) Improve PPP in more complex environments, (2) Effective cooperation with neighbor caching by sharing the whole prefix tree and even dynamic cache size allocation. (3) Improve the graphic user interface of the CCS that allows for designing a new caching algorithm in CCN.

6. REFERENCES

[1] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R.Braynard, "Networking named content", *Communication of the ACM*, Vol. 55, No. 1, pp. 117-124, Jan. 2012.

[2] G. Li, M. Wang, J. Feng, L. Xu, B. Ramamurthy, W. Li, and X. Guan, "Understanding User Generated Content Characteristics: A Hot-Event Perspective", *IEEE Communications Conference*, pp. 1-5, Jun. 2011.

[3] CCNx protocol. Available: http://www.ccnx.org

[4] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn and S. Moon, "Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems", *IEEE/ACM Transaction on Networking*, Vol. 17, No. 5, pp. 1357-1370, Oct. 2009.

[5] W. Zhang, J. Bi, J. Wua and B. Zhang, "Catching popular prefixes at AS border routers with a prediction based method", *Journal Computer Networks*, Vol. 56 ,pp. 1486-1502, Mar. 2012.

[6] D. J. Watts and P. S. Dodds,"Influentials, Networks, and Public Opinion Formation", *Journal of Consumer Research*, vol.34, pp.441ÍC458, 2007.

[7] V. Chaoji, S. Ranu, R. Rastogi and R. Bhatt, "Recommendations to Boost Content Spread in Social Networks", *International World Wide Web Conference (WWW)*, 2012.

[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", *ACM/IEEE symposium on Architectures for networking and communications systems (ANCS)*, pp. 15-26, Oct. 2012.

[9] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)", *Telecom ParisTech*, Technical report, Paris, France, 2011.

[10] Z. Ming, M. Xu and D. Wangy, "Age-based Cooperative Caching in Information-Centric Networks", *IEEE Conference on Computer Communication Workshops (INFOCOM)*, pp. 268-273, Mar. 2012

[11] Z. Li, G. Simon and A. Gravey, "Caching Policies for In-Network Caching", *IEEE Conference on Computer Communications and Networks (ICCCN)*, 2012.

[12] Cho, K., Lee, M., Park, K., Kwon, T. T, Choi, Y., and Park, S., "WAVE: Popularity-based and Collaborative In-network Caching for Content-Oriented Networks", *Infocom'2012 workshops, pp.316-321,2012.*

[13] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla and J. Wilcox, " Information-Centric Networking: Seeing the Forest for the Trees", *Procedure of 10th ACM Workshop Hot Topics in Networks*, Nov. 2011.

[14] G. Tyson, S. Kauney, S. Miles, Y. El-khatibz, A. Mauthez and A. Taweel, "A Trace-Driven Analysis of Caching in Content-Centric Networks", *IEEE Conference on Computer Communications and Networks (ICCCN)*, 2012.

[15] J. Han, S. Kim, T. Chung, T. Kwon, H. Kim and Y. Choi, " Bundling Practice in BitTorrent: What,How, and Why", *ACM SIGMETRICS*, 2012.

[16] IRCache project, www.ircache.net.