Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 795 – 806
© TÜBİTAK
doi:10.3906/elk-1807-73

Cache pressure-aware caching scheme for content-centric networking

Xi LUO[1,2], **Ying AN**[1,*]
[1]Institute of Information Security and Big Data, Central South University, Changsha, P.R. China
[2]Department of Information Technology, Hunan Police Academy, Changsha, P.R. China

Received: 09.07.2018 • **Accepted/Published Online:** 16.11.2018 • **Final Version:** 22.03.2019

Abstract: Content centric networking (CCN) is a new networking paradigm to meet the growing demand for content access in the future. Because of its important role in accelerating content retrieval and reducing network transmission load, in-network caching has become one of the core technologies in CCN and has attracted wide attention. The existing caching schemes often lack sufficient consideration of node cache status and the temporal validity of user requests, and thus the cache efficiency of the network is greatly reduced. In this paper, a cache pressure-aware caching scheme is proposed, which comprehensively takes into account various factors such as content popularity, cache occupancy rate, cache replacement rate, and the validity period of the *Interest* packet to achieve reasonable cache placement and replacement. Simulation results show that the proposed scheme effectively improves the cache hit rate and the resource utilization while decreasing the average response hops.

Key words: Content-centric networking, in-network caching, cache pressure, temporal validity

1. Introduction

With the development of network communication technologies, the Internet has seen rapid growth in the types of services provided and the amount of data exchanged. The main body of network application has gradually shifted from the initial computing resources sharing to the generation, distribution, and acquisition of content. The traditional host-centric network architecture has been unable to adapt to the needs of the current Internet development. To meet the growing demand for content access in the future, a wide exploration of the future Internet architecture has been carried out. As one of the most representative and promising solutions, content-centric networking (CCN) [1] has received a lot of attention. It takes the content itself as the main body of the network communication and realizes the evolution from the host-to-host communications paradigm to a content-centric one.

In the CCN architecture, users request their desired contents by sending *Interest* packets. When a CCN node receives an *Interest* packet, a match query will be performed based on the name of the content. If the content matching the *Interest* packet is found, the corresponding *Data* packet will be returned along the reverse delivery path of the *Interest* packet. At the same time, ubiquitous in-network caching [2] is adopted in CCN, and the response contents will be cached by the nodes along the delivery path. In this way, the network acts as not only a content transmission carrier but also a content storage carrier. In-network caching allows content copies to be cached at the intermediate nodes, which are closer to the requesting users. It can effectively increase the possibility of providing the nearest response to the potential user requests in the future and greatly

*Correspondence: anying@csu.edu.cn

reduces the latency and bandwidth requirements for content acquisition as well as the access pressure on the original content servers (OCS). Therefore, in-network caching has become one of the research hot spots in CCN.

In recent years, researchers have conducted extensive research on content caching schemes for CCN. Among them, the simplest is the LCE (Leave Copy Everywhere) algorithm, which is adopted as the default cache strategy in CCN [3]. In this algorithm, however, the content is replicated by each intermediate node it passes through, which results in excessive cache replacements and redundant content copies. To solve these problems, a series of improved solutions have been proposed after LCE, which can be roughly divided into two categories: centralized strategy and distributed strategy. Jmal et al. [4] proposed a new centralized caching strategy named OFAM, in which the most popular content will be cached by a special management node called "Cache Management" (CM), instead of caching requested contents at every traversed node. Liu et al. [5] combined the Edmonds–Karp algorithm and the marginal allocation algorithm to develop an efficient centralized caching policy to solve the problem of content placement and service scheduling in femtocell caching networks. In addition, the approaches proposed in [6–9] are all typical centralized caching strategies. In centralized caching strategies, the caching decisions and the storage of content replicas are often overly relying on a small number of special nodes. It leads to excessive consumption of resources of these nodes while the resources of other nodes are underutilized, and consequently the actual performance requirements of the network can hardly be met. Therefore, the distributed caching strategy recently has received more and more attention and has gradually become the main research direction of CCN caching.

Laoutaris et al. [10] proposed an improved caching strategy called Leave Copy Down (LCD), in which the content is only cached by the immediate downstream neighbor of the response node along the forwarding path, but it still cannot effectively reduce the redundant caching on the nodes closer to the OCS. In [11], under the assumption that the content request mode and network topology are predictable, the cache deployment is formalized as a linear programming problem to achieve global optimal performance. However, the demand for network global knowledge seriously affects its practicality in the actual CCN environment. Hu et al. [12] proposed a distributed random cache placement strategy, in which the caching node is randomly selected from the intermediate nodes on the return path of content with a certain probability. To some extent, the random decision strategies reduce the redundancy of cached contents, but they cannot guarantee the optimality of caching node selection. Cho et al. [13] presented a popularity-based caching scheme called WAVE, which provides more caching opportunities to the contents with high popularity. In [14], Domingues et al. proposed a caching placement strategy for multitiered and multidomain hierarchical network systems. In this strategy, each node sets up a reinforced counter (RC), which is associated to each content stored there. The RC associated to a given content is incremented by one at every exogenous or interdomain request to that content, and when the counter exceeds some threshold, the corresponding content will be stored. Moreover, the RC is also decremented at a given established rate. When the counter reaches zero, the corresponding content will be evicted. Similarly, Sirichotedumrong et al. [15] put forward a prioritized probabilistic caching scheme (PPC) to selectively cache more important data more than the others by using heterogeneous caching probabilities. In this scheme, each node evaluates the priority of data according to the influence of the data on the quality of content reconstructed from the data. Then the caching probability can be calculated by considering the popularity and priority of data. Other similar studies include [16–18], in which the caching decisions are made primarily based on the relevant characteristics of content. However, these mechanisms often ignore some important information such as node importance, cache status, and so on, which greatly affects the effectiveness of the algorithms. In other studies, researchers proposed to design caching strategies from the perspective of node-related attributes. In

[19–22], the authors adopted a similar idea, which is to determine the cache location of content by considering the betweenness centrality of node. The betweenness centrality of a node is a relatively stable property, while the network dynamic information such as available cache capacity and load state of the node that plays an important role in caching decisions are not fully considered in this scheme. Therefore, a large number of content replicas are concentrated on the high betweenness nodes, which leads to the excessive resource consumption of these nodes while the resource of the other nodes may not be fully utilized.

Some researchers attempted to improve the performance of caching decision in CCN by synthetically considering the characteristics of nodes and contents [23–26]. For example, Li et al. [24] presented a content popularity and node level matched-based probability caching scheme (MPC). In this scheme, the nodes are classified into different levels according to hop account to the requester, betweenness centrality, and cache space replacement rate. Then the node will cache the content replicas whose popularity matches its node level with a high probability in order to maximize the cache utilization and improve the content diversity in networks. In [25], a lightweight on-the-fly caching scheme named RBC-CC was proposed, which utilizes routing betweenness centrality and weighted popularity considering an aging factor to optimize the selection of the caching node. In this scheme, the popular contents will be cached in the important nodes with high betweenness centrality as much as possible for the purpose of improving cache hit rate and reducing content access delay. Unfortunately, it lacks the consideration of node cache capacity in the caching decision-making process, so the cached contents are easily replaced prematurely before being used to respond to the corresponding user requests due to the cache space limitation. Psaras et al. [27] proposed a probabilistic in-network caching scheme, ProbCache. By jointly considering the cumulative cache capacity on the content delivery path and the distance from the OCS, a utility function is designed to implement caching decision-making. Additionally, the authors in [28] proposed a producer-driven caching scheme, in which content routers can collaboratively decide whether to cache chunks according to the availability of cache capacity and the popularity of the chunks. Although this type of algorithm takes node cache capacity into account, the main problem is that the available cache capacity of a node does not always accurately reflect the differences in cache status among nodes because the cache of most nodes will gradually approach and reach saturation under heavy-load and resource-limited situations.

In view of the above problems in the existing CCN caching schemes, this paper proposes a cache pressure-based selective caching scheme that synthesizes a variety of information such as cache occupancy rate, cache replacement rate, content popularity, and residual subscription time to evaluate the rationality of caching node selection and the cache value of content. Compared with the existing research, our work mainly has two contributions:

(1) We design a cache pressure-based caching node selection algorithm, which utilizes cache occupancy rate and cache replacement rate of a node to evaluate the cache pressure on the node, combining the content popularity to achieve reasonable caching node selection;

(2) An effective cache replacement algorithm is also presented. In this algorithm, each node evaluates the cache value of content using the content popularity and the residual subscription time. Then the content with low cache value will be replaced with priority.

The rest of the paper is organized as follows. Section 2 presents the design details of the proposed cache pressure-aware caching scheme. In Section 3, we evaluate our caching scheme by extensive simulations on NS-3. Finally, we conclude our work with future work in Section 4.

2. Cache pressure-aware caching scheme

The proposed cache pressure-aware caching (CPAC) scheme mainly includes two parts: a caching node selection algorithm and a caching content replacement algorithm. In the caching node selection algorithm, the cache pressure on each node will be calculated first. The *Interest* packets will record the minimum value of the cache pressure on nodes along their forwarding path during the propagation process. When the corresponding *Data* packet returns, the caching node is reasonably selected according to the cache pressure and the content popularity. When the available cache space is insufficient, the node will sort the cached contents by the cache value, which is calculated according to the content popularity and the residual subscription time. Then the content with low cache value will be preferentially discarded.

2.1. Cache pressure estimation

Cache status is an important basis for caching node selection, and the available cache capacity or cache occupancy rate is the most common measure reflecting the node cache status. However, in the case of heavy load, the caches of most nodes are approaching saturation, i.e. the available cache space is close to zero (the cache occupancy rate is close to 100%). At this time, the actual cache status of each node cannot be accurately distinguished by the available cache space or the cache occupancy rate. Considering that newly arrived *Data* packets may trigger replacement of cached contents when the node cache is saturated, we combine the cache replacement rate with the cache occupancy rate as the indicator of node cache status.

For any node v, let $C(v)$ be its total cache capacity. Then its cache occupancy $Occ(v)$ can be defined as the ratio of the total size of the contents currently cached on node v to $C(v)$, namely

$$Occ(v) = \frac{\sum_{i=1}^{n} S(c_i)}{C(v)}, \tag{1}$$

where n is the number of contents currently cached by node v, c_i represents the ith content cached on node v, and $S(c_i)$ represents the size of c_i. The cache replacement rate of node v, denoted by $Rep(v)$, can be defined as the ratio of the replaced contents' size to the total cache capacity of node v in a sampling cycle. The calculation method is detailed as follows:

$$Rep(v) = \frac{\sum_{j=1}^{m} D(c_j)}{C(v)}, \tag{2}$$

where $D(c_j)$ represents the size of the replaced content c_j in node v and m is the number of the replaced contents in unit time. Then we introduce the concept of cache pressure and define it as the sum of cache occupancy rate and cache replacement rate. Specifically, the cache pressure of any node v, denoted by $CP(v)$, is calculated as in Eq. (3):

$$CP(v) = Occ(v) + Rep(v). \tag{3}$$

Thus, when the network load is light (the node cache is not saturated), the replacement rates of most nodes may all be close to zero, and the node's cache pressure is mainly determined by its cache occupancy rate. When the network load is heavy (the node cache is saturated), even if the cache occupancy rate of most nodes approaches 1, node cache pressure can still reflect the difference in cache status of different nodes through the cache replacement rate.

We add a node cache pressure field named CPV to the header of the *Interest* packet to record the minimum value of node cache pressure among all the nodes that the *Interest* packet traverses. The initial value of this field is set to a relatively large value, for example 1000. During the *Interest* packet forwarding process, each intermediate node will compare its cache pressure value with the current record value of the CPV. If the cache pressure value of the node is less than the current record value of the CPV, the smaller value will be written in the corresponding field. Then the lowest cache pressure value among the nodes along the transmission path of the *Interest* packet will be recorded. Finally, the value will be replicated onto the *Data* packets when the requested content is found at some node. Thus, each node in the delivery path can obtain the minimum value of node cache pressure CP_{min} from the *Data* packet.

2.2. Cache placement strategy

When the response *Data* packet returns to the requesting user along the reverse path of the *Interest* packet, the appropriate caching node will be selected according to the cache pressure value of each intermediate node and the content popularity. On one hand, we cache the content replicas as much as possible on the nodes with less cache pressure to ensure the reasonable and balanced use of node resources. On the other hand, the existing research shows that the users usually have obvious content preferences and the content requests follow the Zipf distribution. The popular contents receive more attention and may be requested by more users. Therefore, in the cache decision process, they should get more caching opportunities to meet the greater potential demand for them.

However, content popularity is a global attribute that relates to the total number of requests for content by all user nodes in the network. Unfortunately, the acquisition of global information often requires a huge extra cost in the actual network environment. In order to solve this problem, we first use the content request arrival rate at the node to approximate the local popularity of the corresponding content. Here, the request arrival rate for content c_i at node v is defined as the number of *Interest* packets for content c_i received by node v within a sampling cycle, denoted by $\lambda_v(c_i)$. Then we introduce the concept of popularity weight factor and define the popularity weight factor of content c_i at node v as the ratio of the request arrival rate for content c_i at node v to the total request arrival rate for all contents received by node v, denoted by $\theta_v(c_i)$. It can be expressed by:

$$\theta_v(c_i) = \frac{\lambda_v(c_i)}{\sum_{k=1}^{n} \lambda_v(c_k)}. \tag{4}$$

When the response *Data* packet of content c_i arrives at the intermediate node v, the local threshold of cache pressure at node v, denoted by $CP_{th}(v)$, will be calculated by using the minimum value of node cache pressure recorded in the response *Data* packet and the popularity weight factor of content c_i, namely

$$CP_{th}(v) = CP_{min} \cdot (1 + \theta_v(c_i)). \tag{5}$$

Then node v compares its current cache pressure value $CP(v)$ with this threshold $CP_{th}(v)$. If $CP(v)$ is less than $CP(v)$, node v will be selected as the caching node of content c_i.

2.3. Cache replacement strategy

When a node is selected as the caching node of some content, the corresponding content will be replicated and cached by it. However, if the current available space is insufficient, the cache replacement decision is inevitably needed. In order to ensure the effective use of node cache resources, we evaluate the cache value of content and replace the content with low cache value first to prolong as much as possible the time that the content with high cache value is cached in the network. Here, the most critical problem is how to reasonably estimate the cache value of each content, or in other words how to determine which content is more worthwhile to be cached. As mentioned earlier, the potential user demand for high popularity contents is relatively larger than that for low popularity contents in the future. This means that caching high popularity content is more likely to provide better response services for subsequent user requests. Therefore, from this perspective, the cache value of content should be proportional to the popularity of the content. In our replacement scheme, the popularity of the content is still calculated by the local request arrival rate.

The user request also has a certain validity period. During the validity period, the user will maintain interest in the requested content. However, if the requested content is not obtained in the validity period, the user may lose interest in it. That is to say, each *Interest* packet for some content has a time to live (TTL) property; when the TTL expires, the *Interest* packet will be discarded, which also means that the user request is not valid any more. Since the time for different users to generate *Interest* packets may be different, *Interest* packets for the same content from different requesters may also have completely different residual TTLs (RTTLs). Considering that the shorter the RTTL of the *Interest* packet, the closer the *Interest* packet is to expiring, and the fewer opportunities it can provide response service to the requesters, we calculate an average RTTL based on the RTTLs of *Interest* packets for a certain content at different user nodes and use this average RTTL as another metric for the content cache value. The longer the average RTTL of the *Interest* packet, the greater the cache value of the corresponding content, and vice versa.

Thus, the cache value of any content c_i for a given node v, denoted by $CV_v(c_i)$, is defined as the product of the arrival rate of the request for c_i at node v and the average RTTL of the *Interest* packets for content c_i received by node v, namely

$$CV_v(c_i) = \lambda_v(c_i) \cdot \frac{\sum RTTL_j}{L}. \tag{6}$$

Here, $\lambda_v(c_i)$ is the arrival rate of the request for c_i at node v, L represents the number of *Interest* packets for content c_i received by node v from different users, and $RTTL_j$ represents the residual TTL of the *Interest* packet from the jth user.

When a node makes a cache replacement decision, it calculates the cache value of each content and then replaces the content replica with the lowest cache value first.

3. Performance evaluations

This section presents the simulation results to prove the superiority of CPAC versus the existing caching schemes. The simulations are performed on ndnSIM [29], an NS-3-based simulator specially designed for NDN implementation. We compare CPAC with the other three representative CCN caching schemes: LCE (Leave Copy Everywhere), Betw (a betweenness centrality-based caching scheme), and ProbCache (a probabilistic caching scheme based on cumulative cache capacity). Then the performances of different caching schemes are evaluated with different cache sizes, different content numbers, and different user request patterns, respectively. The major metrics used in our performance evaluation are as follows.

(1) Cache hit rate: It is defined as the probability that a request is responded to by the caching nodes instead of the original content servers. Cache hit ratio is one of the most important metrics used to evaluate the caching performance. The higher the cache hit rate, the smaller the response rate and the load of the original content server and the higher the efficiency of the cache system.

(2) Average response hops: It is the average number of the routers traversed by the response packets from the originator or cache to the requesting router. Average response hops reflects the speed of response to the user's request. A lower average response hops indicates higher access speed as well as higher system efficiency.

3.1. Simulation environments settings

In our simulations, we use the Locality model of GT-ITM to generate a flat random topology that consists of 50 nodes. It is assumed that the user request pattern follows a Zipf distribution with parameter α and the content request arrivals follow a Poisson process intensity λ. Each router is equipped with a cache of the same size and the initial cache state of each node is empty. The *Interest* packet is transmitted by flooding and the default cache replacement policy is LRU. The remaining major experimental parameters are set as given in the Table, except by special declaration.

Table. Experimental parameter settings.

Major parameters	Default	Range
Number of contents	2000	$100 - 5000$
Cache size of node (MB)	100	$5 - 1000$
User request pattern	Zipf: $\alpha = 0.8$	$0.6 - 1.2$
Size of content	1	
Request arrival rate (req/s)	50	

3.2. Simulation results and analysis

To clearly demonstrate the performance difference of the above caching schemes, we first observe the impact of network resources on the cache performance of different schemes by changing cache size and content number, respectively. Then we vary the value of Zipf parameter α to compare the performance of the four schemes in different applications. Simulation results are detailed as follows.

3.2.1. Impact of cache size

In this subsection, we first investigate the impact of varying cache size on the cache hit rate in different caching schemes. The increase of cache space means the improvement of caching capability. That is to say, more contents can be cached in the network for relatively longer periods of time. We can see from Figure 1 that the cache hit rate of each caching scheme grows with the cache size. Among the four caching schemes, LCE is too blind and radical in content cache decision-making. In resource-limited situations, it will inevitably result in a large amount of content redundancy and frequent cache replacement, thereby greatly reducing the cache performance. Therefore, the cache hit rate of LCE is significantly lower than those of the other three. In Betw, node betweenness centrality is used to improve the choice of caching nodes, but it causes the cached contents to be excessively concentrated on the high-centrality nodes. Consequently, the contents cached on these nodes are replaced frequently, thereby affecting the cache hit rate. ProbCache performs probabilistic caching node selection through the estimation of the cumulative cache capacity on the content delivery path. To a certain extent, it achieves a relatively balanced utilization of cache resources, and thus its cache hit rate is higher than that of LCE and Betw. By comprehensively considering the cache pressure on the nodes and the cache value of each content, CPAC can achieve a more sensible cache placement and replacement decision and therefore obtains the highest cache hit rate among the four schemes. As can be seen in Figure 1, when the cache size of the node is 100 MB, the cache hit rate of CPAC reaches 50.5%, which is nearly 8.4%, 44.7%, and 62.3% higher than that of ProbCache, Betw, and LCE, respectively.

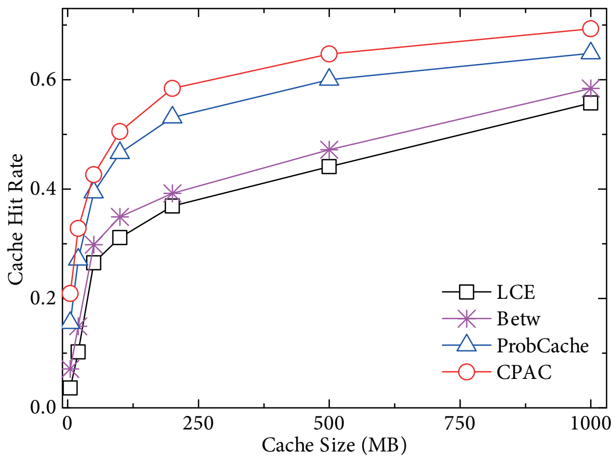

Figure 1. Cache hit rate vs. cache size.

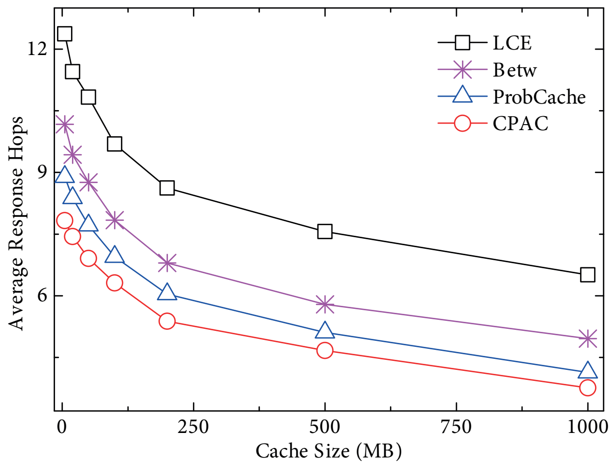

Figure 2. Average response hops vs. cache size.

Similarly, as the node cache increases, the intermediate nodes' caching capabilities are enhanced and each content packet can obtain a longer cache service time. This means that users are more likely to achieve fast content acquisition from the nearest intermediate caching nodes. Therefore, it can be seen from Figure 2 that the average response hops of each scheme gradually decrease as the node cache increases. CPAC chooses to cache the content with high cache value according to the content popularity and the validity period of user requests, which can effectively play a positive role for in-network caching in speeding up the response for potential data access of users. Thus, it remarkably outperforms the other schemes in terms of average response hops. From the figure we can see that the average response hops of CPAC are only 6.31 when the cache size of the node is 100 MB. Compared with ProbCache (6.95), Betw (7.84), and LCE (9.69), it obtains about 9.2%, 19.5%, and 34.9% reduction, respectively.

3.2.2. The impact of content number

Next we analyze the impact of content number on the cache performance of the above listed caching schemes. In the simulations, the node cache size is fixed at 100 MB. In this situation, the greater the content number, the scarcer the cache resources. Therefore, the impact of content number on cache performance is actually another reflection of the relationship between cache size and cache performance.

Figure 3 illustrates the cache hit rate of each caching scheme with varying content number. It can be seen that the cache hit rate of each scheme declines obviously with the increase of content number. The main reason is that the increase of content number leads to an increase in cache replacement, which reduces the probability of users finding their desired contents at the caching nodes. Despite all this, owing to the accurate estimation of node cache pressure, CPAC effectively achieves reasonable selection of caching nodes and balanced utilization of cache resources. Therefore, it still maintains the highest cache hit rate among the four schemes. As shown in the figure, the cache hit rate of LCE is only 57.4% even when the cache space is relatively sufficient (content number is 100) due to its serious waste of resources caused by a large amount of unnecessary caching. By contrast, the cache hit rate of CPAC reaches 76.3%. Even when the content number grows to 5000, CPAC still achieves a 38.1% cache hit rate, which is nearly 14.4%, 77.2%, and 135.2% higher than that of ProbCache (33.3%), Betw (21.5%), and LCE (16.2%), respectively.

From Figure 4, we can see that there is a gradual upward trend in the average response hops of each scheme with the increase of content number. As previously analyzed, this is because the cached contents near the users may be replaced overly quickly when the cache resources are insufficient. It forces the users to obtain the required contents from the distant nodes, resulting in a significant increase of average response hops. However, CPAC is obviously better than the other schemes. When the content number is 5000, the average response hops of CPAC is only 7.55, which is nearly 7.9%, 18.6%, and 34.1% lower than that of ProbCache, Betw, and LCE. This is consistent with the previous analysis of the impact of cache size on cache performance.

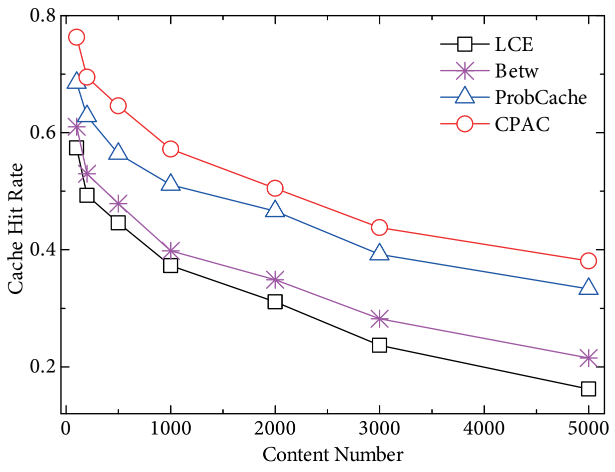

Figure 3. Cache hit rate vs. content number.

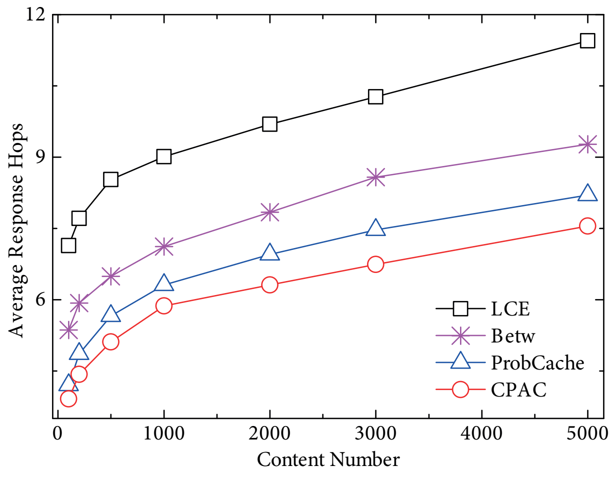

Figure 4. Average response hops vs. content number.

3.2.3. Impact of Zipf parameter α

To further investigate the influence of user access preferences on cache performance, we change the value of Zipf parameter α and observe the performance differences among the four caching schemes listed above in terms of cache hit rate and average response hops.

Generally, a large α value means a high probability that the user requests are issued for popular content, and vice versa. Since all four schemes tend to give priority to the popular content in their cache decision-making process, the cache hit rate of each scheme gradually rises with the increase of α value as shown in Figure 5. While providing caching services preferentially for high popularity contents, CPAC also tries to distribute the cached contents as much as possible to the nodes with relatively sufficient resources and light loads by estimating the cache pressure. Therefore, CPAC distinctly outperforms the other schemes in terms of cache hit rate. For example, when α equals 0.6, the cache hit rate of CPAC is about 40%, which is 105.2%, 75.2%, and 14.1% higher than that of LCE, Betw, and ProbCache, respectively.

From the perspective of the average response hops, it can be seen from the results in Figure 6 that the increase in the value of α leads to an overt decrease in the average response hops. This is because popular contents are given more opportunities to be cached, and as the value of α increases, users' stronger preferences for high popularity contents make more requests be satisfied by the intermediate caching nodes. Hence, the average response hops of each scheme gradually reduce while the content retrieval process is sped up. Similar to the previous experimental results, due to the sophisticated considerations of node cache pressure and content cache value in cache decisions, CPAC obtains the least average response hops among the four caching schemes.

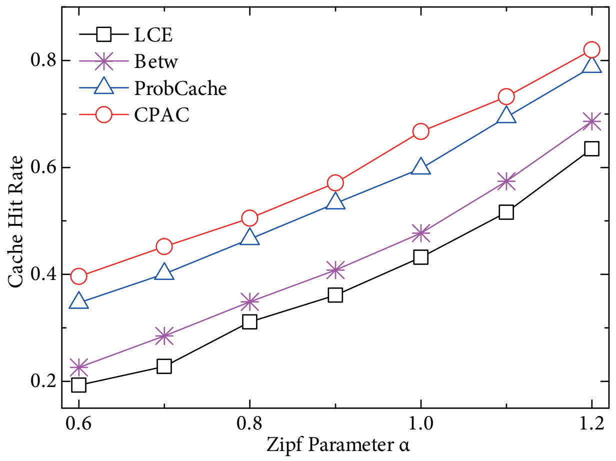

Figure 5. Cache hit rate vs. Zipf parameter α.

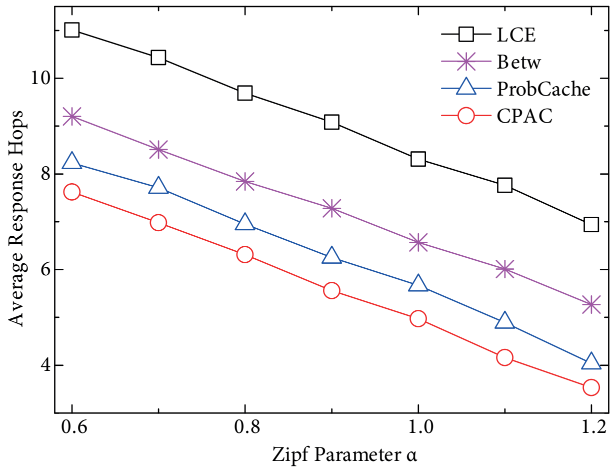

Figure 6. Average response hops vs. Zipf parameter α.

4. Conclusion

In order to fully utilize the network's cache resources, thereby effectively improving the system's cache performance, a cache pressure-based selective caching scheme is proposed in this paper. In this scheme, each node uses cache occupancy rate and cache replacement rate to accurately estimate the cache pressure, and then combines content popularity to optimize the cache placement decision. Furthermore, we comprehensively consider the request arrival rate and the RTTL of the *Interest* packet to evaluate the cache values of different contents, whereby the scheme achieves reasonable cache replacement. The simulation results show that the proposed caching scheme can effectively optimize cache resource utilization and improve the whole cache performance. In the future, we aim to evaluate the performance of the different caching schemes in real-world network topologies with more complex workload models. Meanwhile, we will also extend our algorithm to the mobile network and other complex network environments to further verify its effectiveness.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 61402541 and the Major Science and Technology Special Project of Hunan Province under Grant 2017SK1040.

References

[1] Jacobson V, Smetters DK, Thornton, JD, Plass MF, Briggs NH, Braynard RL. Networking named content. Communications of the ACM 2012; 55: 117-124.

[2] Ahlgren B, Dannewitz C, Imbrenda C, Kutscher D, Ohlman B. A survey of information-centric networking. IEEE Communications Magazine 2012; 50: 26-36.

[3] Laoutaris N, Syntila S, Stavrakakis I. Meta algorithms for hierarchical web caches. In: IEEE 2005 International Performance Computing and Communications Conference; 15–17 April 2004; Phoenix, AZ, USA. Piscataway, NJ, USA: IEEE. pp. 445-452.

[4] Jmal R, Fourati LC. An OpenFlow architecture for managing content-centric-network (OFAM-CCN) based on popularity caching strategy. Computer Standards and Interfaces 2017; 51: 22-29.

[5] Liu T, Abouzeid AA. Content placement and service scheduling in femtocell caching networks. In: Proceedings of the 59th IEEE Global Communications Conference; 4–8 December 2016; Washington, DC, USA. Piscataway, NJ, USA: IEEE. pp. 1-6.

[6] Chanda A, Westphal C. ContentFlow: Adding content primitives to software defined networks. In: Proceedings of the 56th IEEE Global Communications Conference; 9–13 December 2013; Atlanta, GA, USA. Piscataway, NJ, USA: IEEE. pp. 2132-2138.

[7] Chang D, Kwak M, Choi N, Kwon T, Choi, Y. C-flow: An efficient content delivery framework with OpenFlow. In: Proceedings of the 28th IEEE International Conference on Information Networking; 10–12 February 2014; Phuket, Thailand. Piscataway, NJ, USA: IEEE Computer Society. pp. 270-275.

[8] Zhang L, Wang Z, Xiao M, Wu G, Li S. Centralized caching in two-layer networks: Algorithms and limits. In: Proceedings of the 12th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications; 17–19 October 2016; New York, NY, USA. Piscataway, NJ, USA: IEEE. pp. 1-5.

[9] Cui Y, Zhao M, Wu M. A centralized control caching strategy based on popularity and betweenness centrality in CCN. In: Proceedings of the 2016 International Symposium on Wireless Communication Systems; 20–23 September 2016; Poznan, Poland. Berlin, Germany: Springer. pp. 286-291.

[10] Laoutaris N, Che H, Stavrakakis I. The LCD interconnection of LRU caches and its analysis. Performance Evaluation 2006; 63: 609-634.

[11] Borst S, Gupta V, Walid A. Distributed Caching Algorithms for Content Distribution Networks. In: Proceedings of the IEEE INFOCOM 2010; 14–19 March 2010; San Diego, CA, USA. Piscataway, NJ, USA: IEEE. pp. 1478-1486.

[12] Hu Q, Wu MQ, Guo S, Peng L. Random cache placement strategy for content-centric networking. Journal of Xidian University 2014; 41: 131-136.

[13] Cho K, Lee M, Kwon TT, Choi Y, Pack S. WAVE: Popularity-based and collaborative in-network caching for content-oriented networks. In: Proceedings of the IEEE INFOCOM 2012 Computer Communications Workshops; 25–30 March 2012; Orlando, FL, USA. Piscataway, NJ, USA: IEEE. pp. 316–321.

[14] Domingues G, Silva EDSE, Leao RMM, Menasché DS, Towsley D. Enabling opportunistic search and placement in cache networks. Computer Networks 2017; 119: 17-34.

[15] Sirichotedumrong W, Kumwilaisak W, Tarnoi S, Thatphithakkul N. Prioritized probabilistic caching algorithm in content centric networks. In: Proceedings of the 12th International Conference on Computing and Information Technology; 6–7 July 2016; Khon Kaen, Thailand. Berlin, Germany: Springer. pp. 255-265.

[16] Zhang R. Popularity based probabilistic caching strategy design for named data networking. In: Proceedings of IEEE International Conference on Computer Communications; 1–4 May 2017; Atlanta, GA, USA. Piscataway, NJ, USA: IEEE. pp. 476-481.

[17] Zhu X, Wang J, Wang L, Qi W. Popularity-based neighborhood collaborative caching for information-centric networks. In: Proceedings of the 36th International Performance Computing and Communications Conference; 10–12 December 2017; San Diego, CA, USA. Piscataway, NJ, USA: IEEE. pp. 1-8.

[18] Qu H, Xue J, Zhao J. A popularity-based cooperative caching in content-centric networking. In: Proceedings of the 17th International Conference on Communication Technology; 27-30 October 2017; Chengdu, China. Piscataway, NJ, USA: IEEE. pp. 1318-1321.

[19] Chai WK, He D, Psaras I, Pavlou G. Cache "less for more" in information-centric networks (extended version). Computer Communications 2013; 36: 758-770.

[20] Guan J, Yan Z, Yao S, Xu C, Zhang H. The cache location selection based on group betweenness centrality maximization. In: Proceedings of the 12th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness; 7–8 July 2016; Seoul, Korea. Berlin, Germany: Springer. pp. 269-279.

[21] Luo X, An Y. Neighbor cooperation based in-network caching for content-centric networking. KSII Transactions on Internet and Information Systems 2017; 11: 2398-2415.

[22] Kong J, Rui L, Huang H, Wang X. Link congestion and lifetime based in-network caching scheme in information centric networking. In: Proceedings of the 6th International Conference on Computer, Information and Telecommunication Systems; 21–23 July 2017; Dalian, China. Piscataway, NJ, USA: IEEE. pp. 73-77.

[23] Yan H, Gao D, Su W, Foh CH, Zhang H, Vasilakos AV. Caching strategy based on hierarchical cluster for named data networking. IEEE Access 2017; 5: 8433-8443.

[24] Li Y, Zhang T, Xu X, Zeng Z, Liu Y. Content popularity and node level matched based probability caching for content centric networks. In: Proceedings of IEEE/CIC International Conference on Communications in China; 27–29 July 2016; Chengdu, China. Piscataway, NJ, USA: IEEE. pp. 1-6.

[25] Li J, Wu H, Liu B, Fang Z, Zhang S, Shi J. RBC-CC: RBC-Based cascade caching scheme for content-centric networking. Journal of Network and Systems Management 2016; 25: 1-22.

[26] Zhao W, Qin Y, Gao D, Foh CH, Chao HC. An efficient cache strategy in information centric networking vehicle-to-vehicle scenario. IEEE Access 2017; 5: 12657-12667.

[27] Psaras I, Wei KC, Pavlou G. Probabilistic in-network caching for information-centric networks. In: Proceedings of the 2nd ICN Workshop on Information-Centric Networking; 17–19 August 2012; Helsinki, Finland. New York, NY, USA: ACM. pp. 55-60.

[28] Majeed MF, Dailey MN, Khan R, Tunpan A. Pre-caching: a proactive scheme for caching video traffic in named data mesh networks. Journal of Network and Computer Applications 17; 87: 116-130.

[29] Mastorakis S, Afanasyev A, Moiseenko I, Zhang L. ndnSIM 2: An Updated NDN Simulator for NS-3. NDN-0028, Revision 2. Los Angeles, CA, USA: University of California, 2016.