A Cache Replication Strategy Based on Betweenness and Edge Popularity in Named Data Networking

1st Quan Zheng
Laboratory for Future Networks
University of Science and Technology of China
Hefei, China
qzheng@ustc.edu.cn

2nd Yuanzhi Kan
Laboratory for Future Networks
University of Science and Technology of China
Hefei, China
kan@mail.ustc.edu.cn

3rd Jiebo Chen
Laboratory for Future Networks
University of Science and Technology of China
Hefei, China
cjb0725@mail.ustc.edu.cn

4th Song Wang
Department of Automation
University of Science and Technology of China
Hefei, China
wsong@ustc.edu.cn

5th Hongliang Tian
R&D Center
ZTE Shanghai
Shanghai, China
tian.hongliang@zte.com.cn

***Abstract*—Caching content in routers is the most significant feature of Named Data Networking (NDN) and therefore the cache performance is increasingly being concerned for NDN deployment. The default cache policy of NDN is Leave Copy Everywhere (LCE) that leaves the copies in each node the data packet passed, and most of the copies will not be requested again, which leads to the waste of cache resources. The Betweenness Strategy caches data in the node with the maximal betweenness value, which causes a high replacement rate. Considering the betweenness of nodes and the popularity of content, as well as the filter effect of cache, we propose Betweenness and Edge Popularity strategy (BEP) which caches the most popular content in the most important nodes. We also conduct comprehensive simulations based on ndnSIM. By evaluating BEP and other cache replication strategies on a virtual topology, it is indicated that BEP can achieve higher performance in terms of the cache hit ratio, server load and average delay.**

***Index Terms*—Named Data Networking, Caching, Node centrality, Popularity, Cache decision, Network edge, Cache replication strategy**

I. Introduction

Cisco VNI predicted that Global IP traffic will increase nearly threefold over the next 5 years, and video traffic will occupy an increasing proportion [1]. As an example of ICN [2], Named-Data Networking(NDN) [3] caches the content in in-network routers, which decreases the average delay, reduces the server load, and minishes the bandwidth usage. Namely, NDN exchanges precious time and bandwidth with progressively cheaper storage space.

The default cache policy of NDN, LCE [4](Leave Copy Everywhere), caches content on all passing nodes. This policy generates a large number of redundant copies of content, and most of the copies will be replaced before the second request arrives, which leads to the waste of resources. The Betw strategy [5], proposed by W. Chai et al., caches content on the node with the maximal betweenness value. But Betw neglects the high replacement rate in the important node, which causes popular content to be evicted prematurely [6].

Most researches on NDN network caching only take into accounting the characteristics of the content or topology and lack of the combination of the both [5]–[16], which causes the problem of cache redundancy and inefficiency. Though the CPRL strategy [17] considers the both, it ignores the filter effect on the request of the edge node [18]. If a strategy combines with the features of content and topology, it can cache the content more rationally. Based on this, we propose a new cache policy, Betweenness and Edge Popularity(BEP), with the consideration of the betweenness of nodes and the popularity of content, as well as the filter effect of cache. This policy caches the most popular content on the most important nodes, which makes the full use of scarce cache resources and avoids a large amount of cache redundancy. BEP also solves the problem of the high replacement rate and improves the performance of the entire cache system. In this paper, we use betweenness to measure the importance of the node and betweenness will be introduced in section 2.

The rest of the paper is organized as follows. In section 2 we introduce and analyze the related work of NDN cache. In section 3 we describe the algorithm of the BEP strategy and give a specific example. In section 4 we conduct simulations to make comparisons among different cache replication strategies and evaluate the effectiveness of the BEP strategy. We conclude in section 5.

This work was supported in part by the National Key R&D Program of China under Grant SQ2018YFF010138, in part by the National Natural Science Foundation of China under Grant 61233003, and in part by the ZET Technology Cooperation Program.

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

II. Related work

As an important feature and key technology of NDN, cache is one of the research hotspots of NDN networks. Aiming at the shortcomings of the NDN caching strategy, many caching strategies [5]–[17] have been proposed.

The NDN cache is mainly divided into a cache placement strategy and a cache replacement strategy. The former is a strategy for deciding which node or nodes the content is placed on, and the latter is a strategy for retiring which content if there is an another cache request after a node is full. This paper focuses on cache placement strategy. In recent researches of cache placement strategy, researchers are more inclined to use the information of node or nodes for collaborative caching which is divided into explicit collaboration and implicit collaboration. Explicit collaborative cache placement strategies, such as [7]–[9], etc., reduce the content redundancy, ensure the diversity and improve the benefits. However, explicit collaboration requires additional resources of information interaction and computing. Different from explicit collaboration, implicit collaboration does not need to obtain the state information of nodes in the network in advance. The main implicit collaborative cache placement strategies are Betw [5], MCD [11](Move Copy Down), LCD [12](Leave Copy Down), ProbCache [11], etc..

The default cache placement strategy for NDN is LCE [4], which caches content on all nodes that the packet traversed. The LCE algorithm is simple and efficient, but it will cause a lot of content redundancy [7].Most of the cached content will be evicted before the secondary request, thus becoming an invalid cache.

The LCD strategy [12] caches the content on the next hop of the hit node each time, that is, once for each hit and the content moves one hop to the user. This strategy implicitly uses the content request frequency to move content with a high request frequency to a node close to the user, and solves the problem of high redundancy of LCE. Because of its slower moving speed, however, content is likely to be replaced during the move and the user has to regain the content from the upstream node. The MCD strategy [11] is basically the same as the LCD, except that the MCD strategy will delete the copy from the upstream node after the content moves downstream. MCD is less redundant than LCD [11]. But once the content is replaced, the user will have to re-acquire it from the source server in MCD condition.

The probabilistic caching [13], considered as a simple and efficient strategy, caches the content with probability p when the content passes through each node, which solves the problem of high redundancy of LCE to a certain degree. Nevertheless, probabilistic caching is based on a random and uncertain mechanism, which means that it is difficult to find a certain way to optimize to achieve a further improvement. In addition, the limit of the implement conditions is also a bottleneck [13].

The CLCE [10](Conditional Leave a Copy Everywhere) strategy ensures that the content is cached in the node where the content is locally and recently popular. This strategy caches the content quickly in the entire network, but the node caching the content is not always the core one of the network and the main network flow does not always pass this node. Hence, the CLCE does not guarantee the high hit ratio in entire network.

The ProbCache strategy [14] proposed by Psaras et al. comprehensively considers the available cache size of the node and the distance between the content and user. First, the closer to the user node, the greater the probability that the content will be cached. Second, the nodes with sufficient cache capacity have a higher probability to cache the content. However, the ProbCache strategy may cause cache competition among the edge nodes, resulting in a higher replacement rate and thus, a lower hit ratio.

The Betw strategy [5] proposed by Chai et al. utilizes the concept of Betweenness [19] in a complex network. Betweenness is a measure of the importance degree of the node and the definition, respectively, is given by:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \tag{1}$$

Where V is the set of all nodes in the network topology, $C_B(v)$ is the betweenness of node v, s and t are the other two nodes different from v, $\sigma_{s,t}$ is the number of all shortest paths from node s to t, $\sigma_{s,t}(v)$ is the number of the shortest paths from node s to t and passing node v. From the definition it can be noticed that the greater the betweenness, the more paths through the node, and the node will become a hub node in the network. The Betw strategy caches content on the nodes with the greatest betweenness of nodes that the interest packets passed through. Apart from reducing redundancy, Betw allows more other nodes to obtain content quickly using the feature of high betweenness of the caching nodes. The main defect of this strategy is that it does not take into account the high replacement rate brought by this solution [6]. If all content is placed on the node with the highest betweenness, the problem of the high rate of the cache replacement will result from the limitation of the cache capacity of the node. The popular content is replaced and the unpopular content occupies the cache.

Yu et al. proposed a kind of hierarchical caching strategy, CPRL [17], which classified the cache and content according to the number of requests and hops to selectively cache the content. This strategy does not consider the filtering effect on the request of the edge node [18], resulting in inaccuracy classification. Reference [15] considers node popularity and cache replacement rate simultaneously, however, only using the number of the requests as the evaluation criteria of the node's importance will cause the cache to be concentrated around the source node. From the perspective of cache benefit, reference [16] caches high-benefit content with a high probability in a random way. This strategy is an improvement of probabilistic caching [13] and ProbCache [14], but it cannot solve the problem of the cache competition among the edge nodes.

Most of researches on NDN network caching take into account the characteristics of the content or topology, and ignore the filtering effect of the cache. If a strategy can combine with the features of the content and topology, more information will be used to place the network content more rationally, thereby improving the cache hit ratio, reducing the server load and ultimately enhancing the user experience.

III. BEP CACHING PLACEMENT STRATEGY

In order to improve the ratio of cache hit, reduce the server load and decrease the average delay, we propose a BEP(Betweenness and Edge Popularity) cache replication strategy based on the dynamics and popularity of the content, as well as the importance of the nodes in the topology. Periodically, the popularity of each content is counted at the edge of the network, and the cache location of the content is determined by the ranking of popularity and the importance of the nodes along the path. The core algorithm of BEP is to cache the most popular content on the most critical nodes and non-popular content is placed on the secondary nodes, which solves the problem of high replacement rates resulting from the Betw that all content is placed on the core nodes. BEP makes more efficient use of scarce cache resources to achieve better cache effects in typical topologies of the local ISP networks. The specific algorithm is described as follows:

A. Statistics of edge dynamic popularity

Che et al. proposed in [20] that a cache node acts as a low-pass filter in the tree topology for the LRU(Least Recent Used) strategy. The node caches high-frequency content and low-frequency content will be forwarded through this node. Therefore, the count of the requests for the upstream nodes cannot truly reflect the request frequency of all content in the tree topology. The calculation of the content popularity with this frequency cannot be representative and generally correct.

Since the typical topology of the operator in the edge network is a tree topology, we count and calculate the popularity of the content in the edge nodes, that is , the leaf nodes.

The popularity of the content is based on the number of user requests for the content. The algorithm counts the number of requests for each content in each leaf node in cycles and calculates the corresponding popularity. The formula for calculating the popularity of content C in node n in the ith clock cycle is as follow:

$$Popularity_{c,n}(i) = \alpha \times Popularity_{c,n}(i-1) + (1-\alpha) \times N_{c,n}(i) \quad (2)$$

Where α is an attenuation factor, $0 < \alpha < 1$, indicating the proportion of the content popularity before the current cycle. $N_(c,n)(i)$ represents the total number of times the content C has been requested in the ith clock cycle of the node n. $Popularity_{c,n}(0)$ is set to 1 at the initiation of the popularity statistics. The farther away from the current cycle, the smaller the proportion in the popularity calculating. This method is an implementation of the iterative algorithm, which can dynamically adapt to changes in the popularity of the content and better reflect the recent popular content. It is an accurate judgment basis for the cache object.

To achieve statistics and calculations of dynamic popularity, we attach a popularity table to the NDN network node, as shown in Table I. The popularity table is shared across all edge nodes. Assuming that the value of α is 0.75, the historical popularity of the content and the number of times the content is requested in the current cycle are known. The current popularity of the content can be calculated by formula (2).

TABLE I
POPULARITY TABLE OF EDGE NODES

Content	**Historical popularity**	**Request times in current cycle**	**Current popularity**
/video/1.mpg	25.24	30	26.43
/video/2.mpg	16.32	20	17.24
/video/3.mpg	14.23	10	13.17
/video/4.mpg	9.16	0	6.87
...	...	...	...

B. BEP caching algorithm

In the network topology, different nodes have different degrees of importance. There are more connections with others for the important nodes. Thus we believe that these nodes are more crucial in the topology and more network requests will pass through them. If the most popular content is placed on the most important nodes, a significant portion of requests will be responded in important nodes and no more forwarding will occur, which can improve the overall network hit ratio and reduce the server load.

Guan et al. [21] proposed that the betweenness centrality has the better performance than other centrality in the CCN network. Therefore, we choose the betweenness centrality as a metric of the importance of the node. The BEP strategy can be summarized briefly as follows: The popularity of the content is calculated at the edge node, and then the target betweenness is figured out at the responding node according to the ranking of the popularity of the content and the size of the cache. In the return path of the data packet, the node to cache the content is selected according to the target betweenness.

The interest packet generated by users is forwarded to the server on the basis of the corresponding forwarding policy (Algorithm 1). At the first hop, namely, the edge node, the BEP strategy obtains the popularity ranking of the content according to the content popularity table and attaches it to the interest packet. In the forwarding process of the interest packet, BEP records the betweenness of all nodes along the forwarding path in the form of the betweenness array which is attached to the interest packet. When the cache is hit or the interest packet reaches the server, the responding node computes $R_{betweenness}$ according to the following formula:

$$R_{betweenness} = \left\lfloor \frac{Rank_c}{CacheSize} \right\rfloor \quad (3)$$

Where $R_{betweenness}$ is the ranking of the betweenness of the selected node in the betweenness of all nodes along the forwarding path, $Rank_c$ is the ranking of the popularity of the content C, and $CacheSize$ is the size of the node. Then the corresponding betweenness is selected from the betweenness array according to $R_{betweenness}$ and attached to the returned data packet.

For example, assume that $R_{betweenness}$ is 2 ($Rank_c$ is 2 and $CacheSize$ is 1) and the betweenness array is [0.7, 0.3, 0.4, 0.1]. The final betweenness we get is 0.4, because the element 0.4 is the second largest element in the betweenness array.

In the return process (Algorithm 2), the betweenness in the data packet is compared with that of the nodes along the back-forwarding path. When the two is equal, the data is cached in the current node.

It should be noted that if the calculation result of Formula 3 is larger than the length of the betweenness array, the data will be forwarded directly to the user without comparison. In other words, this content will not be cached in the cache network.

Algorithm 1 BEP: node v receives an interest packet

initialize $(rank = 0, betw = 0)$

1: **if** data in cache **then**
2: get rank r of the content from the popularity table;
3: compute $R_{betweenness}$;
4: get the betweenness array $betws[]$ from the interest packet;
5: figure out a specific value $betw$ according to r and $betws[]$;
6: attach $betw$ to the data packet;
7: send the data packet;
8: **else if** node v is server and provide data **then**
9: get rank r of the content from the popularity table;
10: compute $R_{betweenness}$;
11: get the betweenness array $betws[]$ from the interest packet;
12: figure out a specific value $betw$ according to r and $betws[]$;
13: attach $betw$ to the data packet;
14: send the data packet;
15: **else**
16: **if** node v is an edge node **then**
17: count and compute the popularity rank r of the content;
18: attach r to the interest packet;
19: **end if**
20: get node v' betweenness $betw$;
21: attach $betw$ to the interest packet;
22: forward the interest packet to the next hop;
23: **end if**

Algorithm 2 BEP: node v receives a data packet

initialize $(betw = 0, betw_v = 0)$

1: get the betweenness $betw$ from the data packet;
2: get node v' betweenness $betw_v$;
3: **if** $betw$ equals $betw_v$ **then**
4: cache the content of the data packet;
5: **end if**
6: forward the data packet to the next hop;

Because the betweenness is only related to the topological structure, all the betweenness used in this paper is calculated beforehand and stored in each node. The following is a specific example to introduce the BEP strategy.

As shown in Fig. 1, there are three user nodes and one server node. Assume that the cache size of the intermediate five network nodes is 1. With formula (1) we can figure out the value of the betweenness of each node. The numbers below the nodes in Fig. 1 represent the respective value of the betweenness. At the initial moment, the cache of the nodes $v1 \sim v5$ is empty. If user A requests the content of the name '/video/1.mpg' at this time, the edge node $v3$ obtains the popularity rank of the content, 1, by inquiring in the popularity table after receiving the interest package. Hence $v3$ sets the $Rank_c$ of the interest packet to 1 and appends the value of the betweenness, 0.25, of itself to the betweenness array of the interest packet. Then the interest packet is forwarded to the node $v2$. After receiving the interest packet, the node $v2$ appends its own betweenness value of 0.71 to the betweenness array. Finally, the interest packet reaches the server S, and S obtains the $Rank_c$ value and betweenness array [0.61, 0.71, 0.25]. Then we can figure out that $R_{betweenness} = 1$ by using $CacheSize = 1$ and formula (3), so the maximum value in the betweenness array, 0.71, is attached to the returned data packet. In the return process, the betweenness value in data packet is compared with that of each node along the path, and when the betweenness value is equal to 0.71, the data is cached in the current node. In this case, the content is cached in the node $v2$. If user A requests the content of the name '/video/2.mpg', then $R_{betweenness} = 2$, and the returned data packet will carry the second largest betweenness value 0.61, so the node $v1$ will cache this content.

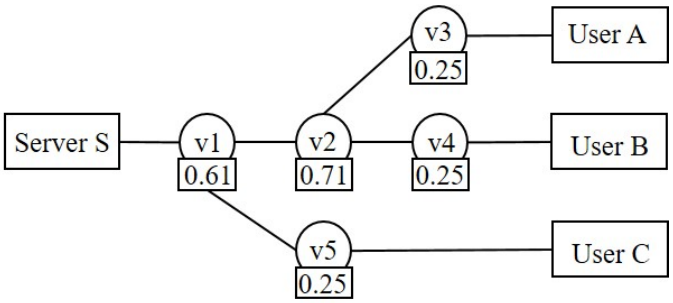

Fig. 1. Topology of the specific example.

IV. Performance evaluation

A. Experimental setup

We now evaluate our replacement strategy through simulations on ndnSIM [22]–[24]. Experimental environment: MacOS High Sierra 10.13 operating system, 4G memory, Intel Core i5 16GHz CPU and 2.4 version ndnSIM. The topology of simulation adopts a tree structure with 7 layers. The child nodes are generated from the root node randomly and each node has 0 to 5 child nodes. Finally, 628 nodes are generated, including one source server node and 377 user nodes. There are $N = 2000$ different contents in the network and each content is 1024KB. Content popularity follows a Zipf-Mandelbrot distribution [25] with a parameter ranging from 0.5 to 1.5, and requests from user nodes follow a Poison distribution with the rate λ which equals 10 req/s in our experiment. The cache size of each node is the same and is C. The value range of C is 10-100, which controls the ratio between the cache size of the CS table and the total content from 0.005 to 0.05 and ensures that the cache capacity of the node is much smaller than total content to meet the real situation of the network. When counting the edge popularity in the BEP algorithm , the attenuation factor is 0.75, the statistical calculation period is 1 second, and the simulation duration is 50 seconds. We choose the first 5 seconds as the system initial warm-up phase and the data generated in the last 45 seconds as the experimental data. We also realize 5 cache placement algorithms commonly used in NDN as the control groups which are LCE, LCD, Betw, Probability, and ProbCache, where Probability has a cache probability of 0.5.

B. Evaluation metrics

We introduce the following metrics as comparative evaluation criterion in our experiment:

1) Hit ratio p_{hit}: Assume that the number of hits of interest packages outside the server node is hit, the number of total interest packages is $total$. Thus we have

$$p_{hit} = \frac{hit}{total} \tag{4}$$

The hit ratio intuitively reflects the performance of the cache system. A high hit ratio means that a larger proportion of requests are responded before the requests arrive at the server node. This is one of the important functions of the cache.

2) Server load: The server load is the number of interest packets that the server need to process per second. The larger the value, the busier the server. The server load reflects the pressure on the server. Network delay and packet loss will be caused by the high server load and the insufficient processing ability of the server. Therefore, it is also necessary to reduce the server load as much as possible.

3) Average delay: The average delay reflects the average duration from the user sending a request to the user receiving a response, in milliseconds. The larger the value, the worse the user experience. One significant purpose of the cache is to enhance the user experience. Therefore, the request delay is an important indicator of the cache system.

C. Analysis of results

We compare the above three performance indicators by changing the cache size of the node and Zipf parameters.

1) Hit ratio: Fig. 2 is a plot of the entire cache hit ratio as a function of cache size under six different caching strategies, with a Zipf parameter of 0.7. From the figure, we notice that the BEP strategy is much better than default LCE of NDN. The total hit ratio of BEP is 55.25% higher than that of LCE when the cache size ratio is 0.025. We also notice that LCD has a higher cache hit ratio than that of the Betw with the small size of the node and has a lower cache hit ratio than that of Betw with the large size of the node. This is because Betw's cache resources are strained when the cache is small, and the disadvantage of its high replacement rate becomes particularly prominent. Therefore, Betw performs poorly. With the increase of the node cache size, a single node can cache more content, which relieves the problem of Betw's high replacement rate.

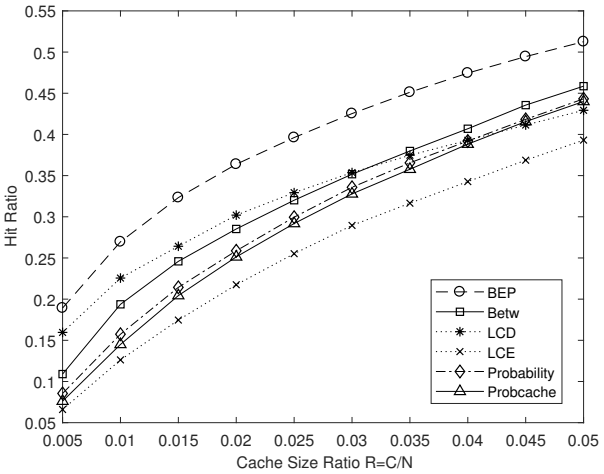

Fig. 2. Hit ratio vs cache size ratio, fix Zipf parameter to 0.7.

As shown in Fig. 3, we fix the cache size ratio to 0.025 and change the Zipf parameter. It's noticed that the BEP strategy is still the best of the six strategies. As the Zipf parameter grows, the users' requests are more focused on less content, so when the Zipf parameter reaches 1.5, most of the popular requests can be cached. The hit ratio of the six strategies are all round 90%, while BEP is as high as 92.14%.

2) Server load: Fig. 4 shows the change of the server load when the Zipf parameter is 0.7 and the cache size of the node increases. In the comparison between LCD and Betw, LCD is still better when the cache size is smaller and Betw is better when the cache size is larger. When the cache size ratio is 0.025, the server load of BEP is 17.30% lower than that of LCE, 10.27% lower than that of Betw, and 9.44% lower than that of LCD.

Fig. 5 shows the change of the server load when the cache size ratio is 0.025 and the Zipf parameter increases. When the Zipf parameter is large enough, the server load can reach a fairly low level. When the Zipf parameter is 1.5, the BEP

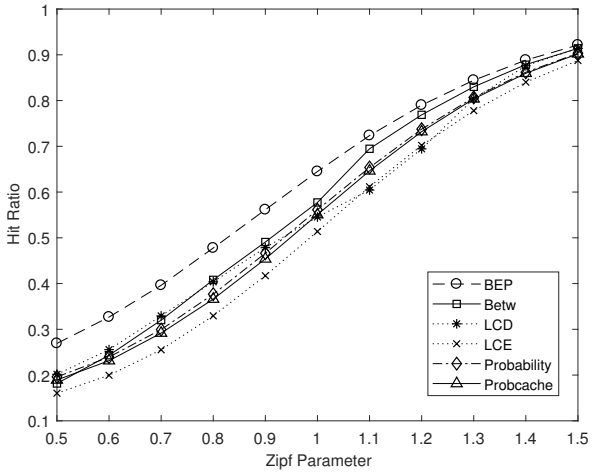

Fig. 3. Hit ratio vs Zipf parameter, fix cache size ratio to 0.025.

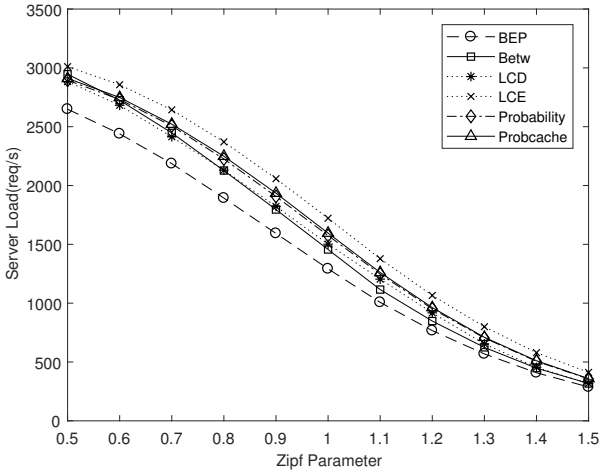

Fig. 5. Server load vs Zipf parameter, fix cache size ratio to 0.025.

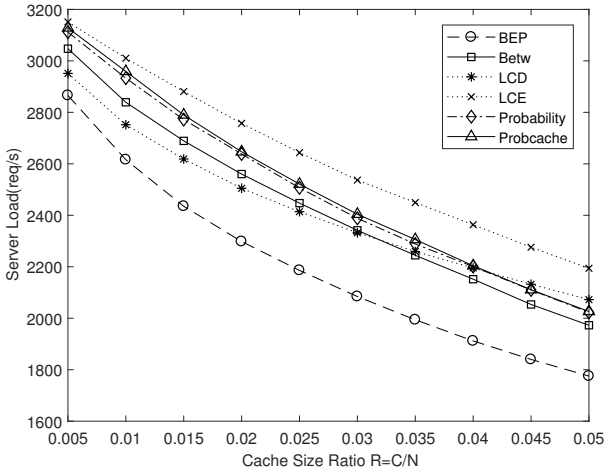

Fig. 4. Server load vs cache size ratio, fix Zipf parameter to 0.7.

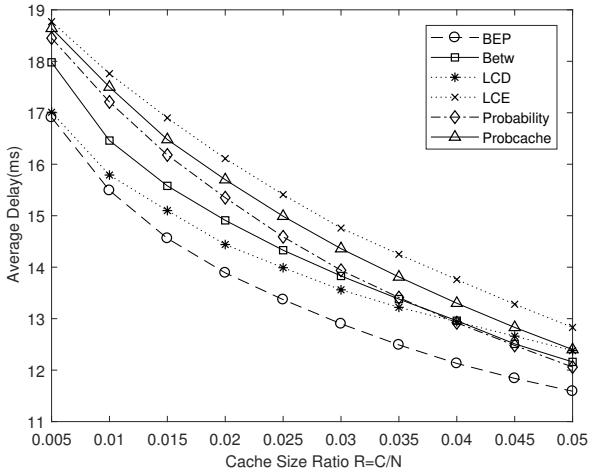

Fig. 6. Average delay vs cache size ratio, fix Zipf parameter to 0.7.

server load is only 287.5 req/s, and the server load of the other five strategies is also within 420 req/s. Overall, the BEP strategy is still optimal.

3) Average delay: Fig. 6 shows the change of the average delay when the Zipf parameter is 0.7 and the cache size of the node increases. Fig. 7 shows the change of the average delay when the cache size ratio is 0.025 and the Zipf parameter increases. From these two figures, we notice that the BEP strategy outperforms the other five strategies. When the Zipf parameter is 0.7 and the node cache size is 0.025, the average delay of BEP is 13.24% lower than that of LCE, 6.70% lower than that of Betw and 4.43% lower than that of LCD.

As shown in Fig. 7, when the Zipf parameter is greater than 1.1, the BEP strategy is almost the same as the Betw strategy in terms of average delay. This is because, when the Zipf parameter is larger, more requests are concentrated on less content. BEP places the most popular content on the most important nodes, and Betw places all the content on the most important nodes. When the Zipf parameter is large enough, almost all users' requests are concentrated on the popular content, so BEP and Betw are almost similar in performance.

V. Conclusion

Based on the analysis of the main problems of some popular NDN caching strategies, we propose the BEP strategy, which integrates the centrality of the node, the popularity of the content and the filtering effect of the cache on the requests. By combining content information and topology information, the BEP strategy overcomes the problem that the hit ratio is not high enough due to the high replacement rate in the Betw strategy. Moreover, the BEP strategy avoids the redundancy and inefficiency of the default policy LCE. Simulation experiments show that, first, BEP is superior to other strategies under various indexes. Second, the performance of BEP and Betw are almost similar when the Zipf parameter is large enough.

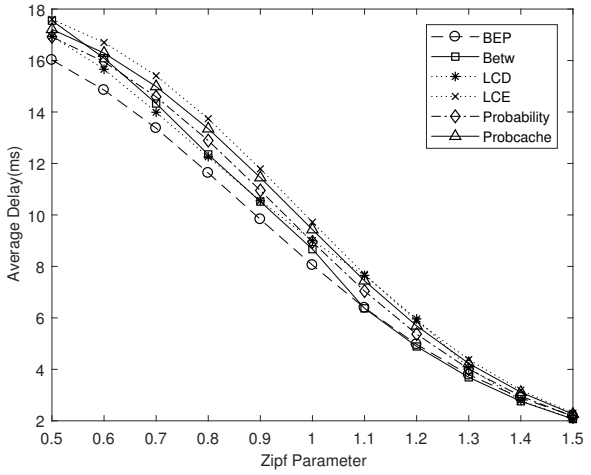

Fig. 7. Average delay vs Zipf parameter, fix cache size ratio to 0.025.

On the basis of this paper, we will provide a local ISP or CP-friendly caching strategy by using more information of the network which contains caching placement strategies and routing mechanisms to reduce caching costs and increase caching benefits on the next step.

REFERENCES

[1] V. Cisco, "Cisco visual networking index: Forecast and methodology 2016–2021.(2017)," 2017.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, July 2012.

[3] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, vol. 157, p. 158, 2010.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[5] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks (extended version)," *Computer Communications*, vol. 36, no. 7, pp. 758 – 770, 2013.

[6] X. D. Cui, L. Jiang, H. Tao, J. Y. Chen, and Y. J. Liu, "A novel in-network caching scheme based on betweenness and replacement rate in content centric networking," *Journal of Electronics & Information Technology*, vol. 36, no. 1, pp. 1–7, 2014.

[7] A. S. Gill, L. D'Acunto, K. Trichias, and R. van Brandenburg, "Bid-cache: Auction-based in-network caching in icn," in *2016 IEEE Globecom Workshops (GC Wkshps)*, Dec 2016, pp. 1–6.

[8] Z. Li and G. Simon, "Time-shifted tv in content centric networks: The case for cooperative in-network caching," in *2011 IEEE International Conference on Communications (ICC)*, June 2011, pp. 1–6.

[9] M. Aoki and T. Shigeyasu, "Effective content management technique based on cooperation cache among neighboring routers in content-centric networking," in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, March 2017, pp. 335–340.

[10] M. Bilal and S. G. Kang, "A cache management scheme for efficient content eviction and replication in cache networks," *IEEE Access*, vol. 5, pp. 1692–1701, 2017.

[11] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013, information Centric Networking. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613002235

[12] N. Laoutaris, H. Che, and I. Stavrakakis, "The lcd interconnection of lru caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609 – 634, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531605000611

[13] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, and Y. Ji, "Performance of probabilistic caching and cache replacement policies for content-centric networks," in *39th Annual IEEE Conference on Local Computer Networks*, Sept 2014, pp. 99–106.

[14] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 55–60. [Online]. Available: http://doi.acm.org/10.1145/2342488.2342501

[15] Y. Ding, Q. Zheng, C. Guo, and S. Wang, "Cooperative caching for icn based on heat and cache replacement rate of node," *Computer Engineering*, 2018.

[16] H. Wu, J. Li, J. Zhi, Y. Ren, and L. Li, "Design and evaluation of probabilistic caching in information-centric networking," *IEEE Access*, vol. 6, pp. 32 754–32 768, 2018.

[17] M. Yu, R. Li, Y. Liu, and Y. Li, "A caching strategy based on content popularity and router level for ndn," in *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, July 2017, pp. 195–198.

[18] C. Williamson, "On filter effects in web caching hierarchies," *ACM Trans. Internet Technol.*, vol. 2, no. 1, pp. 47–77, Feb. 2002. [Online]. Available: http://doi.acm.org/10.1145/503334.503337

[19] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977. [Online]. Available: http://www.jstor.org/stable/3033543

[20] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, Sept 2002.

[21] J. Guan, W. Quan, C. Xu, and H. Zhang, "The location selection for ccn router based on the network centrality," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 02, Oct 2012, pp. 568–582.

[22] A. Afanasyev, I. Moiseenko, L. Zhang *et al.*, "ndnsim: Ndn simulator for ns-3," *University of California, Los Angeles, Tech. Rep*, vol. 4, 2012.

[23] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim 2.0: A new version of the ndn simulator for ns-3," *NDN, Technical Report NDN-0028*, 2015.

[24] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnsim: An open-source simulator for ndn experimentation," *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 19–33, Sep. 2017. [Online]. Available: http://doi.acm.org/10.1145/3138808.3138812

[25] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 1, March 1999, pp. 126–134.