Copyright ⓒ2017 KSII

Neighbor Cooperation Based In-Network Caching for Content-Centric Networking

Xi Luo[1, 2] and Ying An[1]
[1] Institute of Information Security and Big Data, Central South University
Changsha, China
[e-mail: anying@csu.edu.cn]
[2] Department of Information Technology, Hunan Police Academy
Changsha, China
[e-mail: hnluoxi@163.com]
*Corresponding author: Ying An

Received October 19, 2016; revised January 25, 2017; accepted March 11, 2017; published May 31, 2017

Abstract

Content-Centric Networking (CCN) is a new Internet architecture with routing and caching centered on contents. Through its receiver-driven and connectionless communication model, CCN natively supports the seamless mobility of nodes and scalable content acquisition. In-network caching is one of the core technologies in CCN, and the research of efficient caching scheme becomes increasingly attractive. To address the problem of unbalanced cache load distribution in some existing caching strategies, this paper presents a neighbor cooperation based in-network caching scheme. In this scheme, the node with the highest betweenness centrality in the content delivery path is selected as the central caching node and the area of its ego network is selected as the caching area. When the caching node has no sufficient resource, part of its cached contents will be picked out and transferred to the appropriate neighbor by comprehensively considering the factors, such as available node cache, cache replacement rate and link stability between nodes. Simulation results show that our scheme can effectively enhance the utilization of cache resources and improve cache hit rate and average access cost.

Keywords: Content-Centric Networking; in-network caching; load balance; neighbor cooperation

This research was supported by the National Natural Science Foundation of China under Grant Nos. 61402541, and the Scientific Research Fund of Hunan Provincial Education Department under Grant No. 15B127.

https://doi.org/10.3837/tiis.2017.05.005 **ISSN : 1976-7277**

1. Introduction

With the rapid development of Internet technology and the popularity of its application, the network requirements have changed from computing resource sharing into the access to mass information. Content service has gradually become the main body of network service. To better cope with the Internet usage shift from sender-driven end-to-end communication paradigm to receiver-driven content retrieval paradigm, some content-oriented network architectures [1-3] are proposed in recent years. As one representative of the novel architectures, Content-Centric Networking (CCN) [4] has attracted wide research interests in the networking community.

In CCN, users are only interested in the actual content itself rather than the hosting server location, and each content object is uniquely identified by its name. There are two packet types in CCN communications, Interest and Data. The user requests content by broadcasting its interest with the corresponding name identifier. If the content which satisfies the interest can be found in the cache of some node, the event is called a cache hit and the node is called a cache hitting node. Then the cache hitting node will send the corresponding data packet back to the requested node following the reverse path of the interest. In order to alleviate the pressure that rapid traffic growth imposes on network bandwidth and enhance the network resource utilization, in-network caching plays a vital role in the CCN architecture and has received tremendous attention from academia recently. CCN provides a ubiquitous in-network caching and the network acts as not only a content transmission carrier but also a content storage carrier. Thus, when a user requests some content, any intermediate node which caches the corresponding content can give a rapid response, without the need to find the original content servers. At present, LCE (Leave Copy Everywhere) [5] is adopted by CCN as the default caching strategy which replicates the content at each node along the downloading path. However, this strategy introduces excessive caching redundancy and decreases the diversity of the whole system's cached contents. Subsequently, a series of selective caching algorithms are proposed [6-9]. These algorithms pick out partial nodes from the network as caching nodes according to certain criteria to reduce caching redundancy and achieve high cache hit rate. Nevertheless, there is a marked disadvantage that a mass of contents are concentrated on a few of important nodes. That is to say, the cache contents are extremely unevenly distributed. It causes some nodes suffer over quick resource consumption and frequent cache replacement, however, the resource of the other nodes are underutilized. As a result, the overall cache performance is degraded.

To solve the above-mentioned problems, the contribution of this paper is as follows.

(1) We illustrate that some existing caching schemes may yield significant cache load imbalance through simulation, and analyze its effect on cache performance.

(2) We propose a Neighbor Cooperation Based In-network Caching (NCBIC) scheme for CCN. In this scheme, the node with the highest betweenness centrality in the content delivery path is selected as the central caching node and the area of its ego network is selected as the caching area. When the central caching node has no sufficient resources, it will transfer part of its cached contents to its neighbor. The aim is to distribute cache load over the whole caching area and relieve cache pressure on the central caching node.

(3) We design a migrated node selection algorithm which selects the appropriate neighbor to cache the migrated content according to the available cache space, the cache replacement rate and the link stability.

(4) We evaluate our scheme through extensive simulation experiments in terms of a wide range of performance metrics. Results demonstrate that it achieves significant performance gains.

The rest of this paper is organized as follows. Section 2 reviews the previous work on caching technology in CCN and introduces our motivation. Section 3 presents a neighbor cooperation based in-network caching scheme. In Section 4, we show the simulation results. Finally, we conclude the paper in Section 5.

2. Related Work

Caching service is one of the important means of efficient content retrieval in CCN, so cache performance optimization becomes a focused issue on CCN caching technology. Generally, it encompasses two major research topics: cache decision strategies and cache replacement strategies. Cache replacement strategies have been widely investigated in the early studies on traditional web caching. Many replacement algorithms have been proposed [10], including Least Recently Used (LRU) [11] which is the most common algorithm in CCN currently and a lot of improved algorithms [12,13]. Due to the demand for line-speed operation of caches in CCN, however, the simplicity of algorithm is more important for cache replacement strategies. In contrast, cache decision strategies become a focus in CCN caching and have attracted considerable scholarly attention. A series of improved cache decision algorithms has been successively proposed after LCE. For example, [14,15] proposed two distributed random cache placement strategies, in which the caching nodes are randomly selected from the immediate nodes along the delivery path. [16] proposed a caching scheme which combined the random cache decision and least recently used (LRU) replacement policy in the content store of CCN. In this scheme, a uniformly distributed random number rand in the range [0, 1] is generated and compared with the configured caching probability pc. If pc is greater than rand, the content is cached in the store and forwarded to the outbound interfaces; otherwise, the content will be just forwarded. Random cache decision strategies can effectively decrease caching redundancy but without considering some important properties such as the distribution of request and the network topology, therefore, they cannot guarantee the overall cache performance of the whole system. Some researchers proposed that the popularity of content can be adopted as a key property for the design of cache decision strategies. Kideok et al. [17] proposed a WAVE policy which adjusts the number of data chunks cached at each node based on their popularity. The upstream node recommends the number of chunks to be cached at its downstream node by setting a cache indication mark in data packet, and the number is exponentially increased as the request count increases. A content popularity and node level matched based probability caching strategy (MPC) is proposed in [18]. In MPC, three parameters including hop account to the requester, betweenness centrality and cache space replacement rate are jointly introduced to evaluate the level of nodes according to the caching property. Then, a probabilistic method to match content popularity to its corresponding node level is adopted to maximize the cache utilization and to improve the content diversity in networks simultaneously. Huang et al. [19] presented a new caching strategy based on content popularity and hops that can save by caching. Interest packets collect useful information on their way to the server and give it to data packets. Finally, every node makes its own decision by considering node location importance and content's popularity brought by data packets. Ming et al. [20] advanced a concept of content age and designed an age-based cooperative caching scheme. In this scheme, each content has an 'age' field which is dynamically adjusted at different nodes according to the content location and popularity. It

aims to decrease publisher load and network delay by spreading popular contents to the network edge. These caching strategies try to push popular contents to the end users so as to improve cache performance. However, replicating contents to the network edge may take a relatively long time and they also fail to reduce caching redundancy from perspective of the whole network. Consequently the effectiveness of these strategies is seriously limited.

In view of the impact of node importance on cache performance, various caching strategies have been explored. For example, Chai et al. [21] presented a centrality based selective caching algorithm—Betw. In this algorithm, betweenness centrality is adopted as a metric of node importance and contents are only cached at the nodes having the highest betweenness centrality along the delivery path. In [22], a novel caching strategy based on nodes' importance to social communities in CCN is proposed. It designed a NodeRank method, which measure the importance of nodes by considering nodes' social attributes. Then the content copies will be placed in the key nodes with high importance scores. These solutions try to shorten the content access delay and achieve high cache hit rate. However, a glaring deficiency of them is that the cached contents are concentrated on a small number of high-centrality nodes, while the resources of low-centrality nodes are underutilized. To illustrate this point, we present a motivating example of Betw as follow. We simulate a 50-node network and each node has a 10MB cache. There are 10000 contents and the content size is 1MB. Given the arrivals of user requests follow the Poisson distribution with parameter λ=100 and the user requests follow the Zipf distribution with parameter α=0.75. We define the cumulative number of cached contents as the sum of the contents which are presently cached in network and the contents have been replaced. It will be used as a metric to reveal the distribution of cache load under Betw. As shown in **Fig. 1**, the cumulative number of cached contents of nodes 4, 9, 17, 28 and 34 are much more than those of the others. It implies that most of the cache load is applied to a small part of nodes. The seriously uneven load distribution will incur excessive replacements and a decrease of cache hits. Consequently, it badly weakens the positive effects of content caching on providing fast response to subsequent requests.

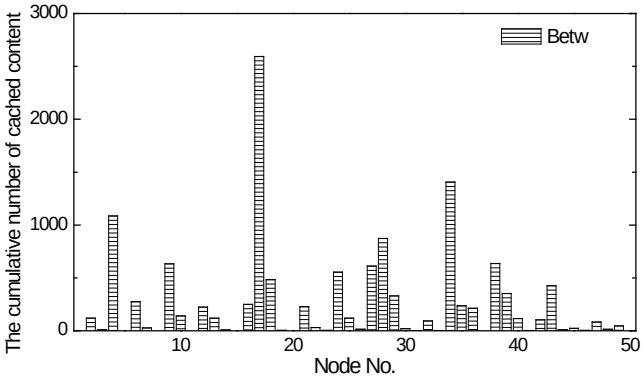


Fig. 1. Cache load distribution of Betw

Some improved schemes have been proposed to solve the above-mentioned issues[23-26]. For instance, in [23], a weight-based probabilistic in-network caching scheme ProbCache is presented. In this scheme, the requested content is cached at each node with a probability. The probability is inversely proportional to the distance from the requester and the current node. The goal of ProbCache is to quickly push the content to the network edge and meanwhile improve cache load balance through the probabilistic caching mode. Due to without

considering the popularity of content, however, there exists fierce resource competitions among different contents at the network edge nodes. As a result, its actual performance is impaired to some extent. In [24], a prioritized probabilistic caching scheme (PPC) is proposed to selectively cache more important data more than the others by using heterogeneous caching probabilities. In this algorithm, each node evaluates the priority of data according to the influence of the data to the quality of content reconstructed from the data. And then, the caching probability can be calculated by considering the popularity and priority of data. However, it makes caching decision simply from the perspective of content, the characteristics of node such as node importance and cache state are not considered. For this reason, its effectiveness is also limited.

In our solution, node centrality is still considered in the selection of caching node. In order to enhance the cache load balance and improve cache hit performance, the node with high centrality is selected to be a central caching node and the area of its ego network is defined as its caching area. Once the cache space of some central caching node is inadequate, the lowest popularity content cached in it will be transferred to one of its neighbor which is selected from its caching area. By doing so, we not only guarantee that the contents are cached on the relatively important nodes as much as possible, but also make full use of the neighbor nodes' resources to optimize cache load balancing. It effectively reduces the cache replacement rate, meanwhile brings a marked improvement in cache hit rate and resource utilization.

3. Neighbor Cooperation Based In-network Caching

It is pointed out in [21] that the relatively important nodes usually have much stronger connection capability than the others, and they have more opportunities to receive the requests from the other nodes. Therefore, it can obtain higher cache hits and faster response time by caching the contents at the important nodes. We agree in the vital effects of node importance on cache performance, and choose the ego network betweenness centrality to evaluate the node importance as well. Unlike Betw, in our scheme, the contents are not always cached at the high betweenness nodes but can be transferred to the other nodes when the resource of the current caching node is insufficient. The goal is to utilize the caching capability of the whole network to achieve cache load balance, and consequently obtain an improvement in cache performance. The work principle of NCBIC is depicted as below.

In NCBIC, each routing node has three main data structures: Forwarding Information Base (FIB), Pending Interest Table (PIT) and Content Store (CS). The FIB records the next-hop forwarding interface toward the potential source of matching content. The PIT is used to store the sources of unsatisfied Interests and each PIT entry includes the content name carried in the interest packet, together with its arrival interface(s). According to the arrival interface, the requested content can be sent back to the user along the reverse path of the corresponding interest packet. When multiple interest packets for the same content are received in succession, only the first interest packet will be forwarded. For the other ones, the node simply records their arrival interfaces in the matching PIT entry and then discards them. Unlike the original design in CCN, we divide the CS into two parts: Common Content Store (CCS) and Content Index Store (CIS). The CCS occupies most part of the cache space and is used to store the full content packets. While only a small part of cache space is assigned to CIS for storing the indexes of contents. Here the index indicates the caching node of certain content and its corresponding access path.

The caching decision procedure of NCBIC includes three parts: 1) cache placement; 2) content migration; 3) cache replacement and update.

In cache placement process, when users send the interest packet to request the content that they want, the nodes on the propagation path of the interest calculate their own ego betweenness values. The interest packet records the highest ego betweenness value among all the intermediate nodes it traverses. The value will be copied onto the data packets when the requested content is found at the original content server or some caching node. On the way to the user, each node matches its own ego betweenness value against the attached one. If the two values match, the current node will be selected as a central caching node, named Caching Center (CC). Meanwhile the ego network of the CC is chosen as a Caching Area (CA). Once a node is chosen as the CC to cache some content, the content is stored in its CCS if it has enough available cache space.

When the occupation ratio of CCS reaches to a critical value, the content migration process will be started. In this process, the CC will filter the cached contents in its CCS to pick a content out and transfer it to the appropriate neighbor node. Meanwhile the CC also creates an index entry in its CIS to record the new cached location information of the content. In this paper, the content transferred away from the CC is called Migrated Content, the node which is selected to accept the migration content is called Migrated Node, and the index information in the CIS is called Migrated Content Index. The content migration process consists of two important algorithms. One is the migrated node selection algorithm, in which the migrated node is chosen from the neighbors of the CC according to cache replacement rate and link stability. Another is the migrated content selection algorithm which is based on the popularity of content. The detailed description of the two algorithms will be depicted in Section 3.1 and 3.2, respectively. The migrated content index contains the migrated content's name, the ID of the corresponding migrated node, the outgoing interface for the migrated node and so on. Based on the content name, once the migrated content index entry which matches some interest is found in the CIS, the CC will forward the interest packet to the specified migrated node through the outgoing interface recorded in the index entry so as to achieve a cache hit.

Finally, when the node has no cache space, the cache replacement and update process will be implemented. In our scheme, the objects to be replaced or updated may involve content packet and migrated content index. In Section 3.3, the handling process will be given in detail.

The communication process of NCBIC mainly includes the following steps:

(1) When an interest packet is arrival, a longest-match lookup is performed on CCS firstly according to the content name. If a matching data packet is found, it will be transmitted out the arrival interface as a response to the interest packet. Otherwise, the node will check its CIS.

(2) If an exact-match CIS entry is found, it means that the desired data packet has been transferred to some migrated node. Then the interest packet will be sent to the migrated node according to the migrated content index. The migrated node will find and return the requested data packet, and then the data packet will be sent back to the requested node according to the corresponding PIT entry.

(3) If there is no match for the interest packet in CIS, then the node will check its PIT. If there is a matching PIT entry the interest packet's arrival interface will be added to the PIT entry. Otherwise, the interest packet will be forwarded to the next-hop node according to its FIB, and a new PIT entry will be created from the interest packet and its arrival interface.

Similarly, when a data packet is arrival, the node first checks its CCS and CIS respectively. If a match is found then the data packet will be discarded. Otherwise, if there is a matching PIT entry then the node will send the data packet back to its original requesters according to the interfaces recorded in the PIT entry. Meanwhile, it will make a decision whether to cache this data packet based on the caching strategy. If no match is found in PIT either, the data packet will be discarded directly.

The pseudo-code is shown in **Table 1**.

Table 1. Algorithm 1: CCMCM Communication Process

Request for content c_i arrives

```
IF c_i is found in CCS THEN
  send back c_i;
ELSE check CIS;
  IF an exact-match CIS entry is found THEN
    send the request to the migrated node according to the CIS entry;
  ELSE check PIT;
    IF a matching PIT entry is found THEN
      add the request's arrival interface into the PIT entry;
    ELSE
      forward the request to the next-hop according to the FIB;
      create a new PIT entry about the request;
    ENDIF
  ENDIF
ENDIF
```

Content c_i arrives

```
CB: the max betweenness of nodes on the c_i's delivery path
CB(v): the betweenness of the current node v
check CCS and CIS;
IF a match entry is found THEN
  discard c_i;
ELSE check PIT;
  IF a matching PIT entry is found THEN
    IF CB(v)==CB THEN
      choose node v to be caching center
      IF freeCache(v) >=size(c_i) THEN
        cache c_i;
      ELSE
        select a neighbor node as migration node N_t;
        select a cached content as migration content M_t;
        migrate M_t to N_t and creat a migration index of M_t on node v;
      ENDIF
    ELSE
      send back c_i to the requesters according to the interfaces recorded in the PIT entry;
    ENDIF
  ELSE
    discard c_i;
  ENDIF
ENDIF
```

3.1 Migrated node selection

One of the major purpose of content migration is to use the caching capability of neighbors to extend the time that the content is cached in network, avoid the content dropping caused by

cache space constrains and improve cache performance as much as possible. In order to achieve the aim, one of the vital issues is to select the appropriate migrated nodes and guarantee the availability of migrated content when a content migration occurs. Two factors are considered in our design of migrated node selection. One is cache state, namely only the node has enough cache space and can keep the migrated content in cache for a relatively long time, will be selected as a migrated node. Another is the connection stability, that is to say, the connection between the migrated node and the CC should have sufficient stability so as to reduce the unavailability of migrated contents due to link interruption. Thus, there is a high probability of getting a cache hit and retrieving the desired content from the migrated node according to the migrated content index.

Considering that the caches of most nodes are approaching saturation when the network reaches a stable state, the available cache size is not a good indicator to accurately reflect the difference of cache state among nodes. Thus, we introduce the concept of cache replacement rate, denoted by $Rep(v)$, which is defined as the ratio of the replaced contents' size to the total cache capacity of node in a sampling cycle. The calculation of $Rep(v)$ is given as Eq.(1).

$$Rep(v) = \frac{\sum_{i=1}^{n} S(M_i)}{C(v)} \tag{1}$$

where $S(M_i)$ is the size of the replaced content M_i in node v, n is the number of the replaced content in unit time, and $C(v)$ is the total cache capacity of node v. Owing to over-quick content replacement may result in cache service unavailability, the contents should be transferred to the nodes with lower cache replacement rate.

Meanwhile, we define the link stability between the CC and any of its neighbors v as the ratio of the link duration to a sampling cycle, denoted by $Stab(v)$. It can be calculated as Eq. (2).

$$Stab(v) = t_c / T_{samp} \tag{2}$$

where t_c is the link duration and T_{samp} is the sampling time. The link between node pairs will disrupt due to node mobility or shutdown. It may cause the cache location information of partial migrated contents in the CC invalid. And then, this part of migrated contents will be inaccessible. In order to promote the availability of migrated content, the nodes that have the higher link stability to the CC should be selected in priority.

According to the basic considerations mentioned above, a weight factor $\gamma(v)$ is designed as a criterion for migrated node selection. It increases proportionately with the link stability but inversely with the cache replacement rate. It is calculated as Eq. (3).

$$\gamma(v) = Stab(v) / Rep(v) \tag{3}$$

When the network is under light load, the cache replacement rates of some nodes may equal to zero due to the available cache space is sufficient or no new content arrives. In this case, we use cache occupancy rate, $Occ(v)$, instead of $Rep(v)$ to calculate the weight factor $\gamma(v)$. Therefore, $\gamma(v)$ should be represented in Eq. (4).

$$\gamma(v) = \begin{cases} \dfrac{Stab(v)}{Rep(v)}, & Rep(v) > 0 \\ \dfrac{Stab(v)}{Occ(v)}, & Rep(v) = 0 \end{cases} \tag{4}$$

When the contents need to be migrated, the CC will calculate the weight factor for all its neighbors. Then the node with the maximum weight factor will be selected as the migrated node.

3.2 Migrated content selection

For the purpose of achieving effective neighborhood cooperative caching, not only the migrated nodes but also the migrated contents should be properly chosen. On the one hand, the migrated contents can be accessed in a short distance (up to 2 hops) between node pairs in a CA. Hence, the redundant caching for a same content in a CA should be avoided as much as possible so as to improve the cache utilization. On the other hand, the more popular the content is, it means that the greater the need of users for it. So caching the contents with higher popularity on the CC is beneficial to improving cache performance. That is to say, the CC should keep the higher popularity contents in its own cache, and move the lower popularity contents to the other nodes when its cache occupancy rate beyond some threshold. The detailed process is described as follows.

(1) Firstly, each node sorts all the contents cached in its CCS in ascending order according to the content popularity.

(2) Then, the CC exchanges the cache contents index with its neighbors for redundancy detection. If the redundant contents are detected between the CC and its neighbors, then the redundancy elimination process will be performed. Specifically, if the redundant contents are less popular (e.g., the popularities are ranked in the bottom 50% among all the contents cached on the CC), they will be removed from the CC directly. Otherwise, they will be discarded by the neighbor nodes. If the redundant contents exist among the neighbor nodes, they will be removed from the node whose cache replacement rate is higher.

(3) We assume that M_c and M_t represent the content with the lowest popularity in the CC and the current migrated node, respectively. Their popularities are represented as $P^c_{\min}$ and $P^t_{\min}$. If $P^c_{\min}$ is not less than $P^t_{\min}$, the CC will choose M_c as the migrated content, and move it to the migrated node to replace M_t. Otherwise, the CC will delete M_c directly.

3.3 Update and Replacement of Migrated Content Index

In NCBIC, the data packets received by nodes can be divided into two types. One is generated from the normal data transmission which is controlled by the content distribution mechanism, and we call this type of packets the normal content packets. Another is the migrated content packets, which is derived from the content migration that triggered by our cache decision mechanism. In order to distinguish the two type of content packets, a flag bit, F_t, is added in the header of each data packet. The initial value of F_t is set to 0, which means that the packet is a normal content packet. While if it is a migrated content packet, the value of F_t will be set to 1. For the normal content packets, the lower popularity contents are replaced in priority when the available cache space is insufficient. For the migrated content packets, however, the update and replacement of the migrated content index also should be considered. Therefore, it will be handled according to different circumstances as follows.

Case 1: the connection between the migrated node N_t and the CC N_c will be broken due to movement or shutdown of N_t.

1) N_t will inform N_c all the migrated contents stored in its cache by sending a cache replacement notification (CRN) which contains the names of all the migrated contents in its cache.

2) According to the CRN, N_c combines all its cached contents and the migrated contents cached at N_t as the pending content collection, and sorts the contents in descending order based on popularity.

3) Then, N_c preferentially chooses the high popularity contents to cache according to its available cache space, and the remaining low popularity contents which its cache cannot contain will be discarded. This means that the migrated contents with higher popularity will be retrieved to replace the lower popularity ones cached at N_c, thereby the popular contents can obtain cache service in priority.

4) When the migrated content is retrieved, its migration flag F_t will be set to 0 and the corresponding migrated content index will be also removed from N_c's CIS.

5) If N_c's cache is full and the popularity of the migrated content is lower than that of any content cached at N_c, the corresponding migrated content indexes will be deleted.

Case 2: due to scarcity of N_t's cache, some migrated content M_i with the lowest popularity need be replaced according to the cache replacement algorithm.

1) N_t sends N_c a CRN which contains the information of M_i.

2) If N_c's cache space is sufficient, the migrated contents will be retrieved immediately. And meanwhile, the same operation as step 4) in Case 1 will be done. Otherwise, N_c will compare the popularity of M_i and each content in its cache.

3) If the popularity of M_i is higher than that of any content cached at N_c, the lowest popularity content will be replaced by M_i. And then N_c reset the migration flag F_t of M_i to zero and remove the index information of M_i from its CIS. Otherwise, M_i will be replaced by N_t and its index information will be deleted from N_c's CIS.

Case 3: the available space of N_c's CIS is insufficient.

1) The migrated content indexes are also replaced according to the priority of content popularity. That is to say, the lower the popularity of the migrated content is, the higher the replacement priority of the corresponding index is.

2) Meanwhile, the migration flag bit F_t will be reset to zero by the migrated node.

4. Performance Evaluations

4.1 Experimental Setup

In order to demonstrate the advantages of our scheme, we compare NCBIC with three representative schemes, namely, LCE (in which caching nodes on the delivery path cache all passing contents), Betw (which only chooses the nodes having the highest betweenness centrality along the delivery path as the caching nodes) and PPC (which selectively caches more important content by using caching probabilities according to content popularity and priority). The simulations are performed on the ndnSIM [27], a NS-3 based simulator which is specially designed for NDN implementation. We use the Locality model of GT-ITM to generate a flat random topology which consists of 50 nodes. The network topological diagram is shown in Fig. 2. In our simulations, we assume that the content popularity follows a Zipf distribution with parameter α and each user generates content requests according to a Poisson process of intensity 10 requests per second. According to [24], the popularity and priority coefficients of PPC are set to 0.9 and 0.1, respectively. By default, the content size is generated using a uniform distribution with values in the range [0.1, 1.9] MB, and the size of each migrated content index is 1KB. In this paper, user requests are forwarded by flooding and LRU is adopted as the cache replacement policy by default. The other experimental parameters are set as given in **Table 2** except special declaration.

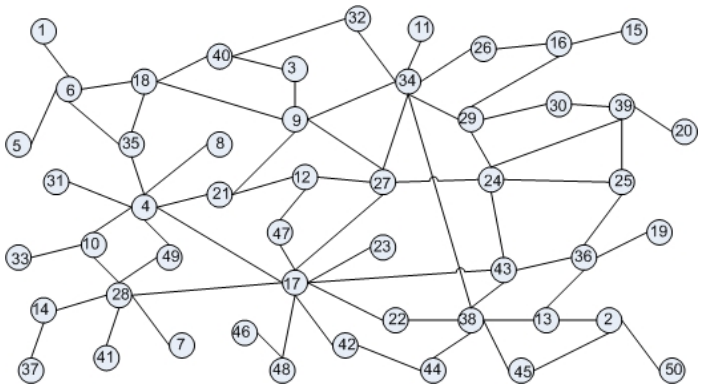


Fig. 2. Simulation topological diagram

Table 2. Parameter settings

Main parameters	Default value	Varying range
Content number	2000	100~3000
Content size（MB）	1	0.1~1.9
Request arrival rate（req/sec）	10	
Cache size（MB）	10	2~100
Zipf parameter α	0.7	0.1~1

The performances of the four schemes are evaluated with different cache sizes, different content numbers and different values of Zipf parameter α, respectively. The following three metrics are used in our performance evaluation:

1) Cache Hit Rate

It is defined as the probability that a request is responded by the caching nodes instead of the original content servers.

2) Average Access Cost

It is defined as the average hop count taken by user interest packets to meet the satisfied content objects.

3) Cache Load Distribution

The difference among different caching decision policies in caching node selection directly impacts the distribution of cache load. To a certain extent, the equilibrium of cache load distribution indicates the network resource utilization. Therefore, we use the cumulative total of cached contents per node (i.e., the times the node is selected as the caching node) to evaluate the effect of different caching schemes on cache load distribution.

4.2 Simulation Results

In order to clearly reflect the performance difference among the four caching schemes mentioned above, we first investigate the effect of network resources through varying cache size and content number. Since the user access patterns differ greatly from the applications, and different user access patterns often have different values of Zipf parameter α. Thus, we then change the value of α to observe the performance of the four schemes in different applications. Finally, the equilibrium of cache load distribution in different caching schemes is discussed. Simulation results are detailed as follow.

4.2.1 Cache hit rate

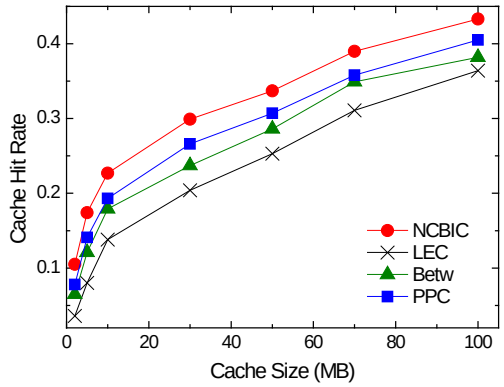
In this subsection, we first compare the cache hit rate with varying network parameters. As shown in **Fig. 3(a)**, the cache hit rate of each scheme increases with the increasing of cache size. This is because the increasing cache space can make nodes keep the cached contents much longer, thus the chance for cache hits increases accordingly. Due to the blindness and radicalness, LCE brings about a large amount of cache replacement and redundancy. Consequently it obtains the least cache hits among the four schemes. Betw optimizes the selection of caching node based on the betweenness centrality, so its cache hit rate is slightly higher than that of LCE. In PPC scheme, each node caches the contents with different probabilities which are dynamically computed according to the popularity and priority. The optimized random caching scheme takes full consideration of content importance and avoids the overconcentration of cache load in some extent. So it obtains the second-best performance. NCBIC not only takes into account node importance in cache decision, but also utilizes the cache resources of neighbor nodes to achieve effective content migration. It relieves the cache pressure on high-centrality nodes and reduces the cache replacement in network. Therefore, NCBIC achieves the best performance on cache hit rate. As can be seen from **Fig. 3(a)**, when the cache size of each node is 10MB, the cache hit rate of NCBIC reaches 22.7%, which is about 64.5%, 26.8% and 17.6% higher than that of LCE, Betw and PPC, respectively.

Fig. 3(b) illustrates the impact of content number on cache hit rate. When the cache size of node remains fixed, cache resources become much scarcer as content number increases. So it can be seen that the cache hit rate of each scheme has a remarkable downtrend with the increase of content number, however, the performance of NCBIC is always superior to the others. This is mainly because NCBIC controls cache redundancy by selecting only one caching node in the delivery path of the content, simultaneously it makes full use of the resource of neighbor nodes to reduce cache pressure and prolong the time that the content cached in network. Due to the redundant caching, the performance of LCE is badly affected. When there are only 100 different content objects in the network, LCE's cache hit rate is about 34.8%, but by contrast the cache hit rates of NCBIC, PPC and Betw reach 71.8%, 69.1% and 66.4%, respectively. Even when the content number rises to 3000, NCBIC still obtains a 20.3% cache hit rate. Compared with PPC (17.6%), Betw (16.1%) and LCE (12.5%), it achieves 15.3%, 26.1% and 62.4% improvement respectively.

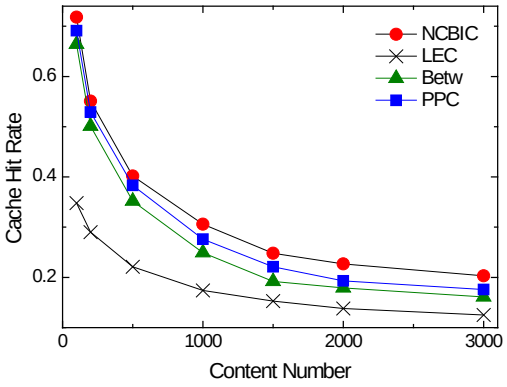
The impact of Zipf parameter α on the cache hit rate is presented in **Fig. 3(c)**. Since the larger the value of α is, the more user requests are issued for the popular contents. Additionally, the cache strategies adapted in all four schemes are inclined to give priority to more popular contents. Thus the cache hit rate of each scheme increases as the value of α increases. Especially for NCBIC and PPC, they provide more caching opportunities to the content with high popularity, so their cache hit rates are increased more markedly with the increasing of α. Owing to the comprehensive consideration of content importance and node state, NCBIC obtains the top performance among the four caching schemes. From the figure, we can see that NCBIC excels the other schemes in cache hit rate. Compared with LCE, Betw and PPC, NCBIC improves the cache hit rate by 152.4%, 73.8% and 58.2% when α is equal to 0.1. Even when α increases to 1, the improvements are also nearly 43.3%, 12% and 3.7%, respectively.

4.2.2 Average access cost

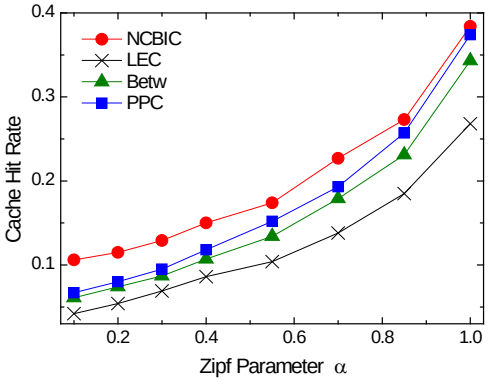
One benefit of in-network caching is to give fast response to user requests through the caching nodes rather than the original content servers. Therefore, the response delay and resource consumption can be reduced accordingly. In this subsection, we evaluate the performance of access cost by using the average number of hops required to find the satisfied contents.



(a) Varying cache size



(b) Varying content number



(c) Varying Zipf parameter α

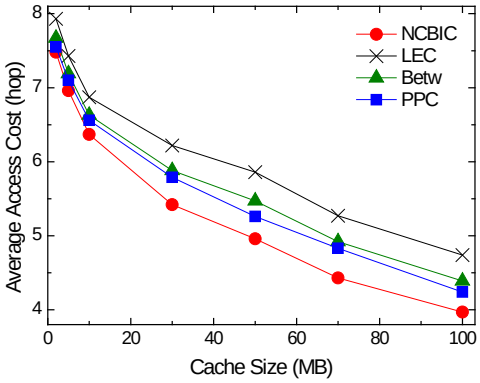
Fig. 3. Cache hit rate with varying network parameters

From **Fig. 4(a)**, we can see that there is a significant decline in the average access cost of each scheme as the increase of cache size. The main reason is the increasing cache capacity enables the nodes to provide cache service for more contents and keep the cached contents for a relatively long time. Owing to the full utilization of neighbor's cache resources, the cache replacements of NCBIC are significantly reduced and the time that the contents exist in the cache is prolonged. Therefore, it obtains the least average access cost of four.

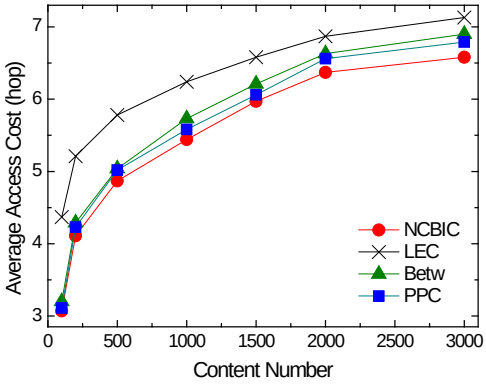
Fig. 4(b) shows the average access cost of each scheme with varying content number. When the number of contents increases, the cache hit rate declines due to the lack of cache resources. It inevitably results in a noticeable rise in the average access cost. Nevertheless, thanks to its

high cache hit rate, NCBIC still obtains the best performance. When the number of contents changes from 100 to 3000, the average access cost of NCBIC is about 15%, 6% and 4% lower than that of LCE, Betw and PPC on average.

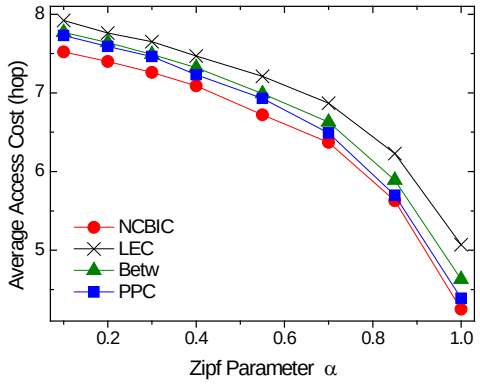
At last, we change the value of Zipf parameter α and observe the variation in average access cost of each scheme as shown in **Fig. 4(c)**. Obviously, the average access costs of all schemes are decreased gradually with the increasing of α. It is in agreement with the prior conclusion that, the increasing preference of user request for popular contents leads to an improvement in cache hit rate and consequently reduces the average access cost.



(a) Varying cache size



(b) Varying content number



(c) Varying Zipf parameter α

Fig. 4. Average access cost with varying network parameters

4.2.3 Cache Load Distribution

In this subsection, we investigate the cache load distributions of different caching schemes. In order to clearly represent the discrepancies on the cumulative number of cached contents per node in different schemes, the content number is increased to 10000 and the request arrival rate is up to 100 requests per second in the following simulations. **Fig. 5** illustrates the cumulative cache load distributions of the four caching schemes during 100 seconds of simulation time. As we can see, since LCE requires every node along the delivery path caching all contents that traverse it, the cumulative number of cached contents per node in LCE is far exceeds that in NCBIC, PPC and Betw. It is a necessary result of excessive caching redundancy. Additionally, the cached contents are unevenly distributed among nodes. Some high centrality nodes carry much more cache pressure than the other one does. Since Betw prefers to choose the node with the highest centrality along the delivery path as the caching node, it results in the most serious inhomogeneity of cache load distribution among the four schemes. In PPC scheme, the content is randomly cached by the nodes in its delivery path with an optimized caching probability. It means that multiple copies of the content may still be left on the delivery path. From **Fig. 5**, we can see that PPC also brings a large amount of caching redundancy. And the cache load is still relatively concentrated on a part of nodes. That is caused by the lack of consideration of node state, although it tries to use the probabilistic caching approach to distribute cache load. By contrast, NCBIC enables the nodes to transfer part of cached contents to their neighbors' caches when necessary. It effectively improves the utilization of network resources while relieving the cache pressure on high centrality nodes. Therefore, the cache load distribution of NCBIC is more even than that of the others. As we can see from **Fig. 5**, NCBIC achieves a more than 50% reduction in the cumulative numbers of the contents cached on the high centrality nodes compared with Betw.

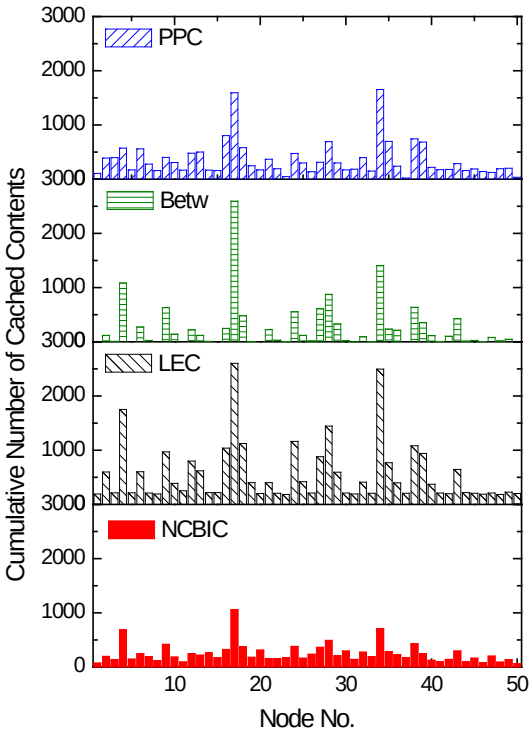


Fig. 5. Cache load distribution in different caching schemes

5. Conclusion

The rational selection of caching node is a key issue in the in-network caching for CCN. In order to achieve efficient utilization of resources and improve cache performance, a distributed cooperative caching scheme is presented in this paper. In this scheme, the betweenness centrality of nodes is considered for optimizing the caching node selection, and furthermore, the effect of cache load distribution is considerably improved by transferring cached contents to neighbor nodes. The simulation results show that the Neighbor Cooperation Based In-network Caching (NCBIC) can effectively enhance the utilization of cache resources and improve cache hit rate and average access cost. In the future, we plan to investigate the efficiency of NCBIC in the context of more realistic applications. Meanwhile, we will extend our algorithm to the mobile network environments.

References

[1] T. Koponen, M. Chawla, B. G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 181−192, October, 2007. Article (CrossRef Link)

[2] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M.Machado, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan and P. Steenkiste, "XIA: An architecture for an evolvable and trustworthy Internet," in *Proc. of the 10th ACM Workshop on Hot Topics on Networks (Hotnets 2011)*, pp. 879-884, November 14-15, 2011. Article (CrossRef Link)

[3] M. Ain, D. Trossen, P. Nikander, et al. "PSIRP D2.3-Architecture definition, component descriptions, and requirements," in *Proc. of the PSIRP 7th FP EU-Funded Project*, 2009. Article (CrossRef Link)

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs and R. L. Braynard, "Networking named content," *Communications of the ACM*, vol. 55, no. 1, pp. 117-124, December, 2012. Article (CrossRef Link)

[5] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical web caches," in *Proc. of the IEEE International Performance Computing and Communications Conference (IEEE IPCCC)*, pp. 445-452, April 15-17, 2004. Article (CrossRef Link)

[6] J. M. Wang, J. Zhang and B. Bensaou, "Intra-as cooperative caching for content-centric networks," in *Proc. of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pp. 61-66, August, 2013. Article (CrossRef Link)

[7] C. Bernardini, T. Silverston and O. Festor, "MPC: Popularity-based caching strategy for content centric networks," in *Proc. of the 2013 IEEE International Conference on Communications*, pp. 3619-3623, June 9-13, 2013. Article (CrossRef Link).

[8] L. Wang, S. Bayhan, J. Kangasharju, "Optimal chunking and partial caching in information-centric networks," *Computer Communications*, vol. 61, no. 5, pp. 48-57, May, 2015. Article (CrossRef Link)

[9] Q. Hu, M. Wu, D. Wang and S. Guo, "Lifetime-based greedy caching approach for content-centric networking," in *Proc. of the 21st International Conference on Telecommunications (ICT)*, pp. 426-430, May 4-7, 2014. Article (CrossRef Link)

[10] S. Podlipnig and L. Böszörmenyi. "A survey of Web cache replacement strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374−398, December, 2003. Article (CrossRef Link)

[11] L. Muscariello, G. Carofiglio and M. Gallo, "Bandwidth and storage sharing performance in information centric networking," in *Proc. of the ACM SIGCOMM workshop on Information-centric networking*, pp. 26-31, August, 2011. Article (CrossRef Link)

[12] D. Rossi and G. Rossini, "Caching performance of content centric networksunder multi-path routing (and more)," *Telecom ParisTech*, Technical Report, 2011. Article (CrossRef Link)

[13] B. Tang, G. Y. Zhang, Z. J. Xing, Y. X. Wu and X. H. Wang, "An Advanced LRU Cache Replacement Strategy for Content-Centric Network," *Applied Mechanics and Materials*, vols. 462-463, pp. 884-890, November, 2014. Article (CrossRef Link)
[14] S. Eum, K. Nakauchi, M. Murata, Y. Shoji and N. Nishinaga, "CATT: Potential based routing with content caching for ICN," in *Proc. of the 2nd ICN Workshop on Information-Centric Networking*, pp. 49−54, August, 2012. Article (CrossRef Link)
[15] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029-1040, April, 2015. Article (CrossRef Link)
[16] J. Garciareinoso, I. Vidal, D. Diez, D. Corujo and L. A. Rui, "Analysis and Enhancements to Probabilistic Caching in Content-Centric Networking," *Computer Journal*, vol. 58, no. 8, pp. 1842-1856, February, 2015. Article (CrossRef Link)
[17] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi and S. Pack, "WAVE: Popularity-Based and collaborative in-network caching for contentoriented networks," in *Proc. of the IEEE INFOCOM Workshop on NOMEN*, pp. 316-321, March 25-30, 2012. Article (CrossRef Link)
[18] Y. Li, T. Zhang, X. Xu, Z. Zeng and Y. Liu, "Content popularity and node level matched based probability caching for content centric networks," in *Proc. of IEEE/CIC International Conference on Communications in China*, pp. 1-6, July 27-29, 2016. Article (CrossRef Link)
[19] S. Huang, Z. Ding and X. Yang, "Caching Strategy Based on Popularity and Saving Content Delivery Hops in Content Centric Networking," *Journal of Computational Information Systems*, vol. 11, no. 20, pp. 7323-7330, October, 2015. Article (CrossRef Link)
[20] Z. Ming, M. Xu and D. Wang, "Age-Based cooperative caching in information-centric network," in *Proc. of the 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1-8, August 4-7, 2014. Article (CrossRef Link)
[21] 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, April, 2013. Article (CrossRef Link)
[22] Y. Cheng, M. Wu, M. Zhao and K. Wang, "Socially-aware NodeRank-based Caching Strategy for Content-Centric Networking," in *Proc. of the 13th International Symposium on Wireless Communication Systems*, pp. 297-303, September, 2016. Article (CrossRef Link)
[23] I. Psaras, W. K. Chai and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. of the second edition of the ICN workshop on Information-centric networking*, pp. 55-60, August, 2012. Article (CrossRef Link)
[24] W. Sirichotedumrong, W. Kumwilaisak, S. Tarnoi and N. Thatphithakkul, "Prioritized Probabilistic Caching Algorithm in Content Centric Networks," in *Proc. of the 12th International Conference on Computing and Information Technology*, pp. 255-265, December, 2016. Article (CrossRef Link)
[25] X. Cui, T. Huang and J. Liu, "Design of in-network caching scheme in CCN based on grey relational analysis," *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 2, pp. 1-8, February, 2014. Article (CrossRef Link)
[26] X. Hu, J. Gong, G. Cheng and C. Fan, "Enhancing in-network caching by coupling cache placement, replacement and location," in *Proc. of 2015 IEEE International Conference on Communications(ICC'15)*, pp. 5672-5678, June 8-12, 2015. Article (CrossRef Link)
[27] S. Mastorakis, A. Afanasyev, I. Moiseenko and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," *NDN-0028*, Technical Report, 2015. Article (CrossRef Link)

Xi Luo is a lecturer in Department of Information Technology, Hunan Police Academy, P. R. China. She received her B.S. and M.S. degree in Communications Engineering from Central South University, P. R. China, and her Ph.D degree in computer science from Central South University, P. R. China. Her general research interests include content-centric networks, in-network caching technology and content distribution.

Ying An is an associate professor in Institute of Information Security and Big Data, Central South University, P. R. China. He received his B.S. degree in automatic control from Central South University, P. R. China, and his M.S. degree in communications engineering from Central South University, P. R. China, and his Ph.D degree in computer science from Central South University, P. R. China. His research interests include information centric networks, delay tolerant networks and optimization of future internet.