Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5648765, 13 pages
https://doi.org/10.1155/2021/5648765

Research Article

Information-Centric Networking Cache Placement Method Based on Cache Node Status and Location

Dapeng Man,[1] **Yao Wang**,[1] **Hanbo Wang**,[1] **Jiafei Guo**,[1] **Jiguang Lv**,[1] **Shichang Xuan**,[1] **and Wu Yang**[1,2]

[1]*Information Security Research Center, Harbin Engineering University, Harbin 150001, China*
[2]*Peng Cheng Laboratory, 518055, China*

Correspondence should be addressed to Wu Yang; yangwu@hrbeu.edu.cn

Received 23 May 2021; Accepted 9 August 2021; Published 15 September 2021

Academic Editor: Dr. Muhammad Shafiq

Information-Centric Networking with caching is a very promising future network architecture. The research on its cache deployment strategy is divided into three categories, namely, noncooperative cache, explicit collaboration cache, and implicit collaboration cache. Noncooperative caching can cause problems such as high content repetition rate in the web cache space. Explicit collaboration caching generally reflects the best caching effect but requires a lot of communication to satisfy the exchange of cache node information and depends on the controller to perform the calculation. On this basis, implicit cooperative caching can reduce the information exchange and calculation between cache nodes while maintaining a good caching effect. Therefore, this paper proposes an on-path implicit cooperative cache deployment method based on the dynamic LRU-K cache replacement strategy. This method evaluates the cache nodes based on their network location and state and selects the node with the best state value on the transmission path for caching. Each request will only select one or two nodes for caching on the request path to reduce the redundancy of the data. Simulation experiments show that the cache deployment method based on the state and location of the cache node can improve the hit rate and reduce the average request length.

1. Introduction

Information-Centric Networking (ICN) is a promising network architecture that is ideal for spreading popular information across the network. Its important feature is the node cache. A copy of the information is deployed to the cache of each router node. Each request to the router node is first searched in the content storage (CS). If there is a request with the same name in the CS, then the router respond with this copy directly. There is no need to fetch information from the content source server, thus reducing network traffic, reducing server bandwidth, and increasing the efficiency of content distribution. Under a reasonable cache deployment strategy and cache replacement strategy, caching will greatly improve the throughput and performance of ICN [1–3].

There are many cache nodes in the ICN. Excessive deployment of cache information on a tremendous number of nodes wastes a lot of storage space and causes data redundancy, which reduces the cache hit rate. Therefore, a good caching deployment strategy is needed to determine which node the cache is deployed. At present, the cache deployment strategy is mainly divided into a noncooperative cache and collaborative cache, and the collaborative cache is divided into an explicit collaboration cache and an implicit collaboration cache. Because in the noncooperative caching strategy, each node does not refer to the state of other nodes in the caching decision process and independently decides whether to cache content. Noncooperative caching will result in a large amount of redundant data in the network cache space, which is of little significance to the limited and valuable cache space [4]. Explicit collaboration caching generally requires a large amount of communication between the cache nodes to determine the best nodes to cache. However, this ignores the impact of network communication and computation on the nodes. The overhead data

communication and calculation will increase the load on the network, which will reduce the overall performance [5–7]. In the implicit collaboration caching, the nodes incurred less overhead during the information exchange of the caching decision. Therefore, among noncooperative caching, explicit collaboration caching, and implicit collaboration caching, implicit collaboration caching has a very high practical significance, is not constrained by the specific circumstances of the network, does not increase excessive network traffic, and reduces caching redundancy [8]. For this reason, this manuscript proposes an on-path implicit cooperative cache deployment method based on the dynamic LRU-K cache replacement strategy. The main contributions of this manuscript are as follows:

(1) Through the dynamic LRU-K strategy, the state of the cache node and the network location are comprehensively judged

(2) Based on the state of the cache node and the location of the cache node in the network, a cache decision-making scheme is proposed in combination with implicit cooperative cache

(3) Improved multipoint caching and decision-making, reducing the repetition of network cache space content, while effectively increasing the cache hit rate

2. Related Work

Caching function is the advantage of ICN. The efficiency of caching directly determines the overall performance of ICN, and the caching deployment strategy of caching is the most important part of the caching strategies. In recent years, scholars around the world have focused more on the performance improvement of ICN brought by cache deployment strategy. For cache deployment strategy, we should focus on efficient network utilization and high availability of data. From the perspective of the P2P system and CDN technology, cache deployment on the edge of the network is very helpful to improve the cache hit rate, but ICN supports the deployment of caches on all routers. Not only the edge nodes but also the central nodes are suitable for the deployment of caches. Therefore, the deployment strategy of caches will be the key factor to increase the overall performance of ICN.

Cache deployment strategy is also called cache decision strategy. At present, there are two classifications of ICN cache deployment strategies in academia. One is classified according to the cache location, which is divided into an on-path strategy and an off-path strategy [9]. On-path strategy means that the cache is deployed on the sending path of interest packages. The specific cache nodes are determined by the cache deployment strategy [10, 11]. The off-path strategy does not depend on the sending path of the interest packet. It determines where the content should exist based on the overall network status or content attributes. The caching method related to the hash method is a typical path-independent caching method [12, 13]. Therefore, the off-path strategy is more suitable for the mobile cache. Another classification method is based on the cooperative approach, which is divided into three categories: noncooperative cache, explicit cooperative cache, and implicit cooperative cache.

The noncooperative caching strategy does not need to rely on the information of other caching nodes and can directly decide whether to cache the content or not. It is a strategy of "single-handedly fighting." This decision-making strategy is relatively simple and does not require excessive judgment. However, the problem that comes with it is that for an ICN with a holistic layout, such a simple cache strategy can affect the overall performance of the network. A typical caching strategy in a noncooperative cache is LCE [14]. In LCE, as the content information is returned, a cache is copied to each cache node passing through the return path. Although such a deployment strategy can simply cache the information content, it also brings high redundancy. The problem of a low overall hit rate has caused a serious waste of resources [15].

The explicit collaborative caching strategy determines the specific cache nodes by adding a centralized controller to the ICN. The centralized controller collects the real-time status of the cache nodes in the network and then makes a judgment. The presence of a centralized controller effectively relieves the pressure on the router. Studies have shown that explicit collaborative caching can greatly improve cache efficiency [16, 17].

In [18], an explicit cache coordination strategy based on HASH is proposed. The HASH value is obtained according to the location, popularity, and history of the cache node. Then, the corresponding cache node is searched for the cache. If this content exists in the cache node, the cached content is sent directly back to the subscriber, and if it does not exist, the subscriber's request is forwarded from this cache node. The HASH-based explicit cache collaboration strategy can quickly locate cache nodes and fast forward requests without generating redundant data.

Currently, the explicit collaboration cache is still under exploration. Although it can project a very good cache deployment strategy, it also has some shortcomings [19]. For example, in a high-speed network environment, each data content needs to pass the controller to help the controller decide which node to cache in; this computing overhead should not be underestimated. Secondly, if the centralized controller is down, it is unfavourable for the network where the centralized controller is located. Whether it hands off the cache node to another centralized controller or discards the current network, there is still a significant impact on overall ICN network.

Although explicit collaboration cache can bring high cache efficiency, it requires a lot of interactive information, and the calculation method is too complicated. The noncooperative cache has a lot of data redundancy. The implicit collaboration cache combines the advantages of the above two cache methods. The implicit collaboration cache mainly relies on some additional messages for cache decisions, such as cache node location, content popularity, probability, and cache node status. Due to the high performance and low cost of implicit collaborative caching, the implicit collaborative

cache has the highest proportion among the three cache deployment strategies. Below are a few implicit collaboration caching strategies.

The main purpose of the LCD [20] (Leave Copy Down) strategy is to keep the cache close to the edge of the network, so the LCD policy will copy a cache on the second cache node of the return path each time, so each cache hit will move this cache once towards the edge node. As the content popularity increases, the cache will get closer and closer to the edge of the network, thus reducing the delay of user access. However, such a caching strategy will leave more copies on the return path with high redundancy. Moreover, for a large number of requests for different information, the cache of the core network nodes will be continuously replaced, and it is not guaranteed the cache hit.

MCD [21] (Move Copy Down) is an optimization of the LCD cache decision strategy. Unlike the LCD, after the cache hit, the cache of the node is deleted, and the redundancy of the information content is reduced. From the perspective of redundancy, MCD has made a very good optimization compared with LCD, but it also has certain disadvantages. Suppose there is a tree network, the root node is a content publisher, the leaf node is a content subscriber, and the other nodes are cache routers [22]. In this tree structure, if a router's cache is frequently hit by a request in a different subtree, the cached location will always be near this node and will not always go to the leaf node.

Prob [23] (Copy with Probability) is a random caching algorithm, sometimes called Bernoulli random caching algorithm, which uses a fixed probability p to make cache decisions. Such a caching algorithm does not guarantee that popular information will be cached. Popular content will have multiple requests, cached with the same probability, and will have more chances to be cached; this algorithm has a strong randomness.

ProbCache [23] proposed improvements based on Prob, and its improvement idea also hopes that the cache exists at the network edge as much as possible. Therefore, its probability is proportional to the distance from the cache node to the content provider (may be a content publisher or a cache node). The number of hops is used here as a measure of distance.

The above implicit cooperative caching schemes effectively reduce the burden of node information exchange of explicit cooperative caching, but they lack the consideration of the network node status and the network location of the caching node. The performance improvement of the caching is limited. In response to the above problems, this manuscript uses the LRU-K strategy to comprehensively judge the state and network location of the cache node. On this basis, this research combines the idea of implicit cache cooperation and proposes a cache decision-making scheme.

3. Analytical Methods

The research on ICN caching strategy can be classified into cache replacement strategy and cache deployment strategy. An effective cache replacement strategy can increase the hit rate of cache nodes. But for global purposes, reducing the average request length and cache redundancy requires a cache deployment policy to manage the global cache nodes [24].

According to the foregoing, implicit collaborative caching is a caching method that is very suitable for ICN. It does not require the global computing and communication capabilities of explicit collaborative caching, nor does it have the large amount of redundant data that noncollaborative caching generates [25]. Typical representatives in implicit collaboration cache are LCD [20], MCD [21], ProbCache [23], etc. These three caching strategies are designed to make the cache closer to the requesting node on the path, thus reducing network latency, but without considering each cache whether the state of the node is suitable to cache the information. Therefore, this paper proposes an on-path implicit cooperative caching algorithm with the following specific objectives:

(1) Improve the cache hit rate, reduce network latency, and reduce the average request length of users

(2) Consider the state and location of the cache node to give the nodes that should be cached

(3) Do not add too many fields to the packet, causing the network packet to be bloated

Based on the above requirements, this paper proposes a cache deployment method based on the status and location of the cache nodes, which considers the position of the cache node on the path, the number of prefiltering queues, and the number of filtering when caching nodes exchange information. It is a bidding strategy that determines the cache location at a very small communication cost [26].

3.1. Concept and Definitions

3.1.1. Cache Node Status Values. The cache node status value is an important indicator for evaluating whether the cache node in the ICN is suitable for caching. The less the cached content is, the more suitable it is the node for the cache. The closer to the user, the more suitable it is the node for the cache. At present, the state value of each cache node is determined by three factors. However, for the cache nodes in different network environments, the proportion of each factor should be different. Therefore, the final state value is calculated by weighting. The formula is as follows:

$$\text{Value} = \alpha \times V_k + \beta \times V_{\text{hop}} + \gamma \times V_{\text{hitk}}, \alpha, \beta, \gamma \in N. \quad (1)$$

Among them, Value represents the final state value, V_k represents the state value based on the number of prefiltered queues K, V_{hop} represents the state value based on the link location, V_{hitk} represents the data of interest packages in the number of prefiltered queues, and alpha, beta, and gamma represent the weight of three state values, respectively. In order to reduce the accuracy problem caused by floating-point representation, the weights of the three state values (α, β, γ) are represented by nonnegative integers.

(1) LRU-K Strategy. The main purpose of LRU-K is to solve the "cache pollution" problem of the LRU algorithm. Its core idea is to extend the "recently used 1 time" criterion to "recently used K times." The K in LRU-K represents the number of recent uses.

Compared with LRU, LRU-K needs to maintain one more queue to record the history of all cached data being accessed. Only when the number of accesses of the data reaches K times, the data is put into the cache. When data needs to be eliminated, LRU-K will eliminate the data whose Kth access time is the largest from the current time.

(2) Cache Queue State. The state of the cache queue is determined by the K value in the dynamic LRU-K algorithm, that is, the number of prefiltered queues. When the number of prefiltering queues of a cache node is large, it means that the number of packets need to be cached by the node is large. Compared with the cache node with a smaller K value, it takes a longer time to put the cache node with a larger K value into the cache queue. Therefore, the function for caching queue state calculation is a monotonic decreasing function.

(3) Link Location. In ICN, it is more desirable that information be cached on edge nodes, which can reduce the average number of request hops for users, the network latency, and the load of the central network. Because the acquisition of the whole network's topology structure is complex and requires a large amount of computation, it does not meet the requirements of implicit collaborative caching, so the hops of interest packages are added to the interest packages as the basis for calculating the relative location of cache nodes in the network. For example, the number of hops of the nearest cache node to the user is 1. If the cache node has no data in the CS, the cache node needs to forward the user's interest package to the cache node with the hop number of 2 until the content publisher or the node with the cache is found. The function of calculating the relative position state value of cache nodes by hops decreases monotonously with the increase of hops, and the higher the hops, the lower the importance.

(4) Location of Prefiltered Queues. In dynamic LRU-K cache replacement strategy, a data packet is put into the cache queue only after the packet has passed the required number of caches in the Kth queue at that cache node. Therefore, when judging whether a cache node is suitable for caching a packet, it should consider whether it is currently in the prefilter queue and which prefilter queue. Otherwise, the packet will appear in the prefilter queue of each cache node, thus underestimating the prevalence of the packet and delaying it from being cached or even failing to be cached. The purpose of this parameter is to reduce the impact of multiple nodes on data filtering and depending on the weight parameter that determines its importance in formula (1); priority should generally be given to nodes where that data is already in the prefilter queue so that it is cached as soon as possible.

3.1.2. Cache Rate. Caching rate is the ratio of the number of caches selected by the node to the number of interest packages received. The concept of caching rate is proposed mainly because when choosing the caching node through the state value, the edge nodes will be frequently selected as the caching nodes. The main reason for this problem is that the distribution of the state value is relatively uneven, and the caching probability of the nodes close to the content source is very small. Therefore, the cache rate is proposed, and the point on the forwarding path of the packet of interest with the smallest cache rate is selected for caching to compensate for the deficiency caused by selecting the cache node only by the state value. The formula for calculating the cache rate is as follows:

$$\text{CacheRate} = \frac{N_{\text{Cached}}}{N_{\text{total}}}. \tag{2}$$

In formula (2), N_{Cached} represents the number of data packets stored in the network cache space. N_{total} represents the number of all data packets in the network space.

3.1.3. Data Packet Status Value. A data packet is a response packet sent back by a content publisher according to the direction of the sending path of interest packages. The data package contains the name and content of the data. However, for the current deployment strategy, it is impossible to know whether the data packages have been cached on the path. Therefore, an additional field needs to be added to tell the subsequent caching node whether the data packages can be cached. When a data packet is cached at a node, the state value of the field should be modified, and subsequent nodes can judge whether to cache the data packet according to the state value. Two nodes, one has the maximum state value, and the other one has a minimum cache rate, need to be found on the transmission path of the packet. There are four cases, which can be represented by 0, 1, 2, and 3, respectively, as shown in Table 1.

3.1.4. Statement Record Table. In order to record the maximum state value and the minimum cache rate of a cache node through which an interest package passes, a Statement Record Table (SRT) is added to each cache node, which contains the data name field and the state value in the interest package. When the data packet returns, firstly querying the state value corresponding to the data name in the SRT table, if there is no record, it shows that the state value is not higher than the maximum state value, and the cache rate is not lower than the minimum cache rate at that time when the interest packet is transmitted, so this node is not selected as the cache node.

The meaning of state value in the SRT table is like Table 2, but it does not need state value 0. If it is not a candidate node of cache nodes, it cannot be stored in SRT. It is not necessary to prepare a state value specifically for this state, and it also reduces the occupied data space, the meaning of state value as shown in Table 2.

For the state value stored in SRT can not directly determine whether it is the maximum state value node and the

Table 1: Data packet status values and implications.

State value	Meaning
0	The maximum state value node has been cached, and the minimum cache rate node has been cached.
1	The maximum state value node is cached, and the minimum cache rate node is not cached.
2	The maximum state value node is not cached, and the minimum cache rate node is cached.
3	The maximum state value node is not cached, and the minimum cache rate node is not cached.

Table 2: State record table state values and implications.

State value	Meaning
1	It cannot be the maximum state value node, it may be the minimum cache rate node.
2	It may be the maximum state value node, not the minimum cache rate node.
3	It may be the maximum state value node or the minimum cache rate node.

Input: Interest Packet (*Pkt*); Statement Record Table (*SRT*)
Output: Operation Statement
1: *Pkt.hop* ⟵ *Pkt.hop* + 1
2: *value* ⟵ get value
3: *cacherate* ⟵ get cache rate
4: *srtstat* ⟵ 0
5: **if***value* > *pkt.maxvalue***then**
6: *pkt.maxvalue* ⟵ *value*
7: *srtstat* ⟵ *srtstat* | 2
8: **end if**
9: **if***cacherate* < *pkt.mincacherate***then**
10: *pkt.mincacherate* ⟵ *cacherate*
11: *srtstat* ⟵ *srtstat* | 1
12: **end if**
13: **if***srtstat* > 0**then**
14: insert *pkt.name*, *srtstat* into *SRT*
15: **end if**
16: forward *pkt*
17: **return***SUCCESS*

Algorithm 1: Interest packet processing algorithm for cache deployment policy.

minimum cache rate node, it needs to be judged by combining the state value in the interest packet. The main reason is that in the process of forwarding the interest package, it is not known whether the next node has a larger state value or a smaller cache rate, but when the interest package is forwarded to the current node, it is the maximum state value or the minimum. When the data packet returns along the route forwarded by the interest packet, if a node encounters the maximum state value, and the state value in the data packet indicates that the node has not been cached at the maximum state value, then the current node is the node with the maximum state value, and the judgment of the node with the minimum cache rate is the same.

3.1.5. Concrete Design. The cache deployment strategy based on the state and location of the cache node selects two nodes to cache in the data transmission path. The first cache node is the one with the best state value, and the second cache node is the node with the lowest cache rate. At present, three factors are affecting the state value. The first one is the status of the LRU prefilter queue, which needs to be given in combination with the dynamic LRU-K algorithm proposed above. The second one is to calculate the location of each cache node on the request path according to the hops of interest packets. The closer to the edge of the network, the higher this value is. The third one is the queue number hit in the prefilter queue. The larger the number is, the more popular the content is, and the more important it is to cache the data packet at this node. Caching on the nodes with the lowest cache rate is to compensate for the nonuniformity of caching based on the state value.

As shown in Algorithm 1, in order not to add additional computing nodes when processing interest packets, the maximum state value and minimum cache rate of the cache nodes are allocated to the interest packets. Then, the state value and cache rate of the current nodes are calculated on each cache node. The larger state value and the smaller replacement rate are recorded in the

Input: Data Packet (*Pkt*); Statement Record Table (*SRT*); Content Store (*CS*)
Output: Operation Statement
1: **if***pkt.datastat* > 0**then**
2: **if***SRT* has *pkt.name***then**
3: *res* ⟵ *pkt.datastat*&*SRT.getStat*(*pkt.name*)
4: **if***res* > 0**then**
5: *pkt.datastat* ⟵ *pkt.datastat* − *res*
6: insert *pkt* into *CS*
7: **end if**
8: remove *pkt.name* from *SRT*
9: **end if**
10: **end if**
11: forward *pkt*
12: **return***SUCCESS*

Algorithm 2: Data packet processing algorithm for cache deployment policy.

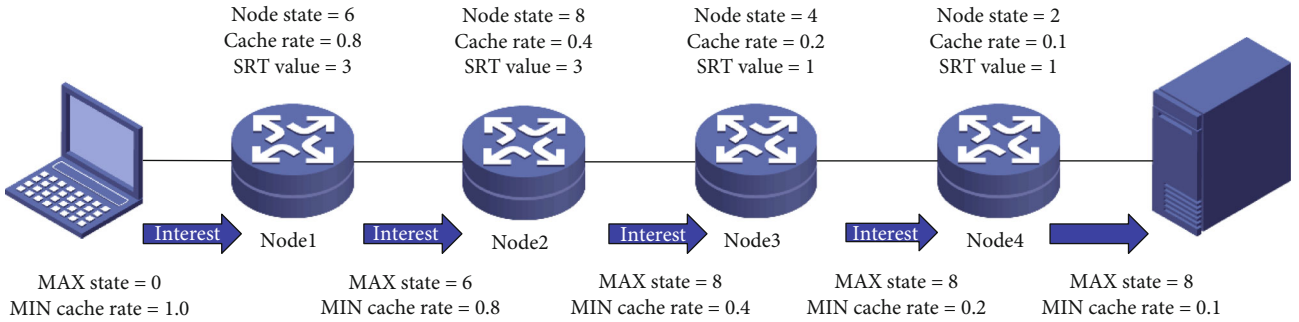

Figure 1: Interest packet forwarding process.

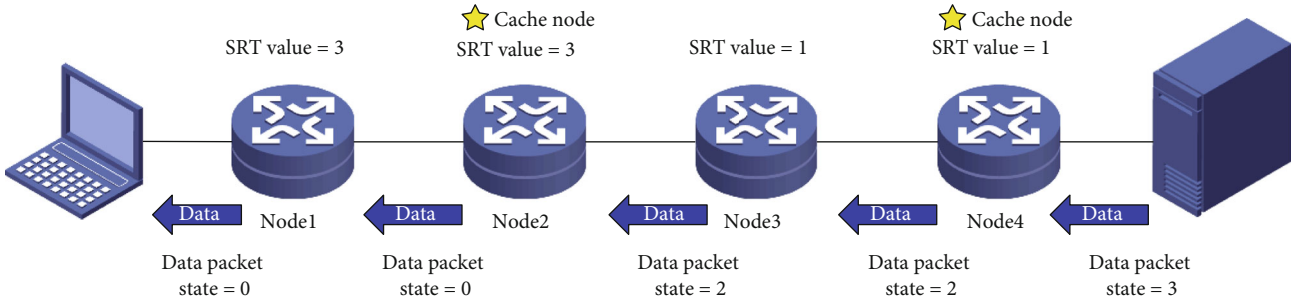

Figure 2: Data packet forwarding process.

interest packages and then insert data name and SRT status value into SRT. A maximum state value field, a minimum cache rate field, and a hop number field need to be added to the interest package. All cache nodes maintain these three fields together. If the state value of a cache node is higher than the maximum state value or the cache rate is lower than the minimum cache rate, the corresponding data is updated, and records are generated in the cache state record table, SRT state values are updated, and the cache state record table SRT is stored on each routing node.

As shown in Algorithm 2, when the cache node receives the response packet, whether the current packet is cached is calculated according to the SRT state value and the packet state value. When the content needs to be cached, the state value is changed so that the subsequent nodes will not cache the packet with the same type, thus ensuring that the data will not be excessively

Table 3: Icarus simulation platform parameters.

Parameter name	Parameter value
Zipf distribution alpha value	0.5-1.2
Number of contents	300000
Total cache size	(0.002-1) * number of contents
Interest packet preheat value	300000
Interest packet measurement	600000

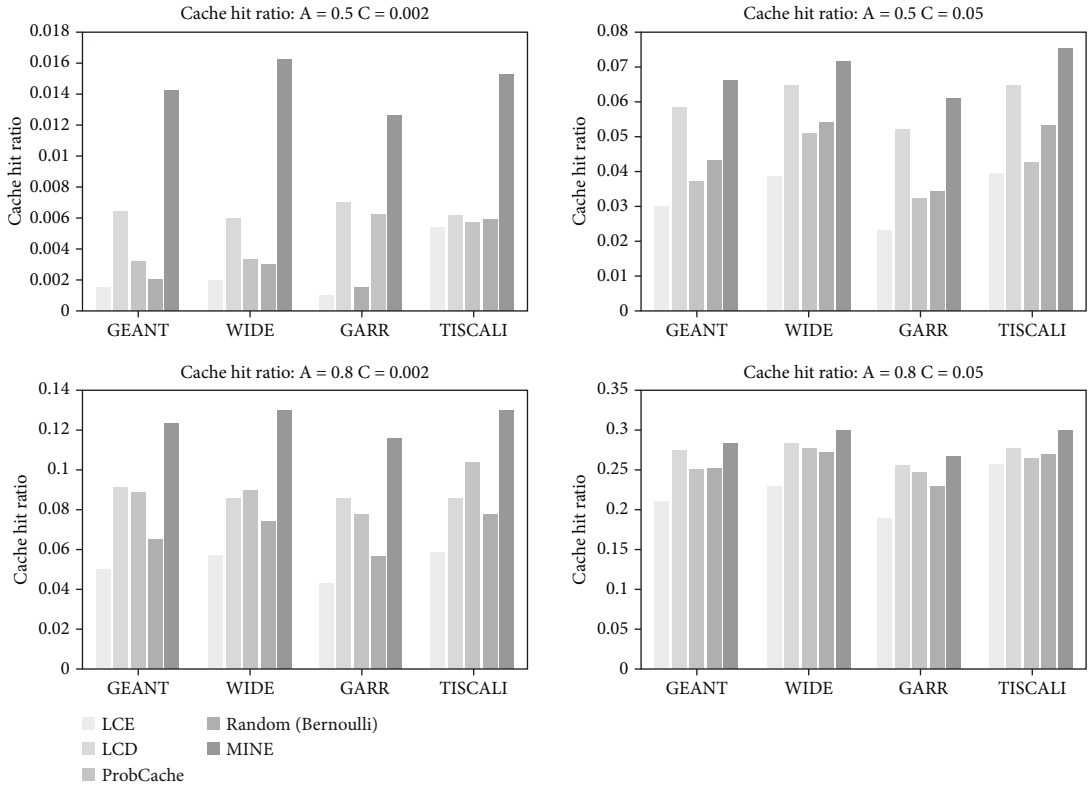

Figure 3: Comparison of cache hit rates.

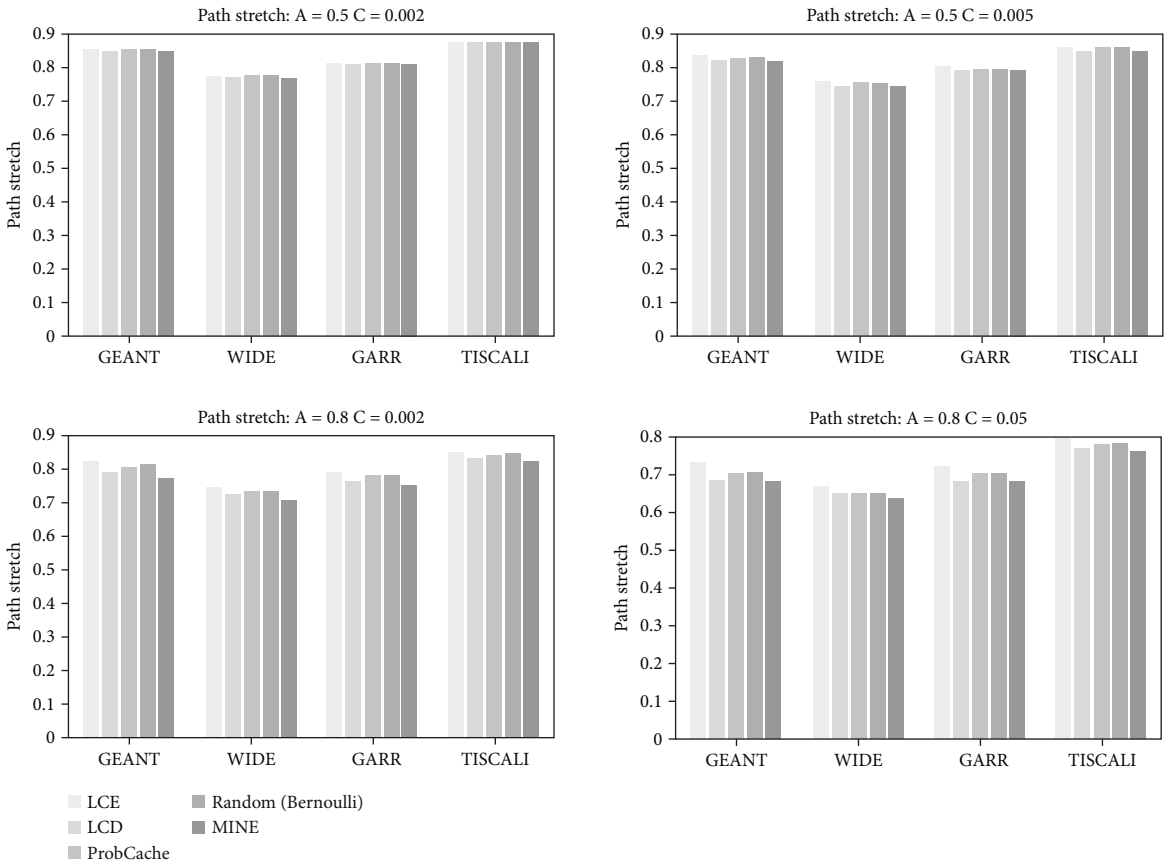

Figure 4: Path stretch comparison diagram.

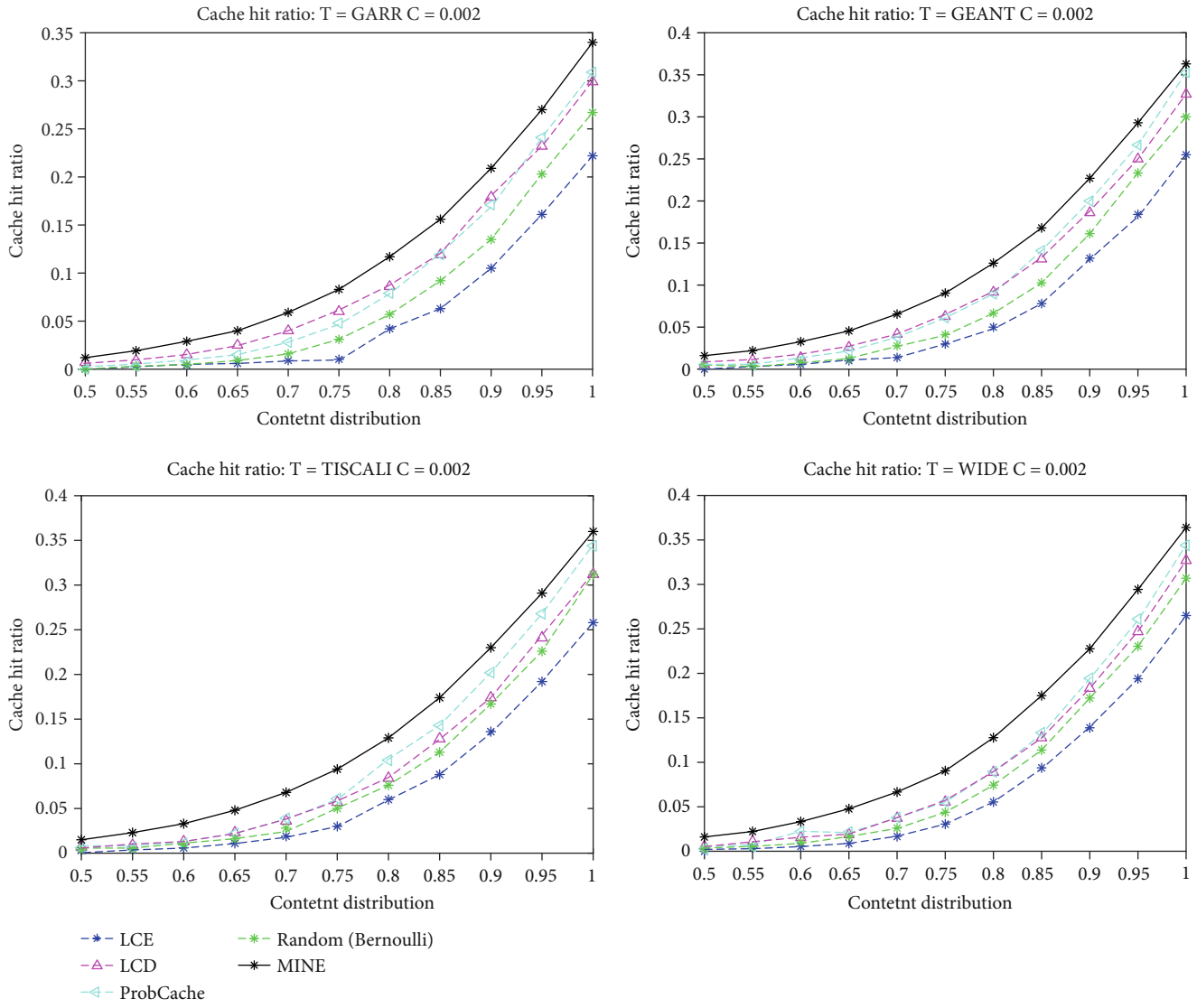

Figure 5: $C = 0.002$. The relationship between cache hit rate and content concentration.

redundant. If no caching is required, the data packet is forwarded directly. After the data packet passes through the cache node, the corresponding records in SRT should be deleted to avoid wasting storage space. In order to cooperate with the implementation of the algorithm, it is necessary to modify the data package accordingly. Add a packet status value field to the packet, the initial value sets the status to 3. In the return path, if the state value of the state record table in a node and the state value of a packet do not result in 1 when doing a logical and operation, then the cache is inserted into the cache of the node, and the state value of the packet is updated. If replacement is required, a dynamic LRU-K cache replacement strategy is executed.

The selection process of cache nodes is shown in Figures 1 and 2. Figure 1 shows the changes of some parameters in the process of forwarding interest packages. Figure 2 shows the selection of cache nodes in the process of packet response. Two graphs show the process of selecting cache nodes at one time.

4. Simulation Results and Analysis

4.1. Simulation Environment. The simulation environment uses the Icarus simulator, version v0.7.0, which can be downloaded from GitHub. The address is detailed in [27]. Icarus is an event-based ICN simulator, implemented in Python by Lorenzo Cerno and others. It is specially developed to evaluate the performance of the cache system in the information-centric network. It uses the MVC (Model-View-Controller) design pattern. This model implements the basic functions of ICN, the view monitors changes on the network model, the controller processes events and responses, and the results act on the network model.

The server used in the simulation experiment is the Sugon A620r-G server. The CPU is AMD Opteron(tm) Processor 6320, 1.4 GHz, 16 core, 32 threads, and 16 GB of memory, and the operating system is CentOS Linux release 7.2.1511. The parameters for the dynamic LRU-K cache replacement policy are set as follows:

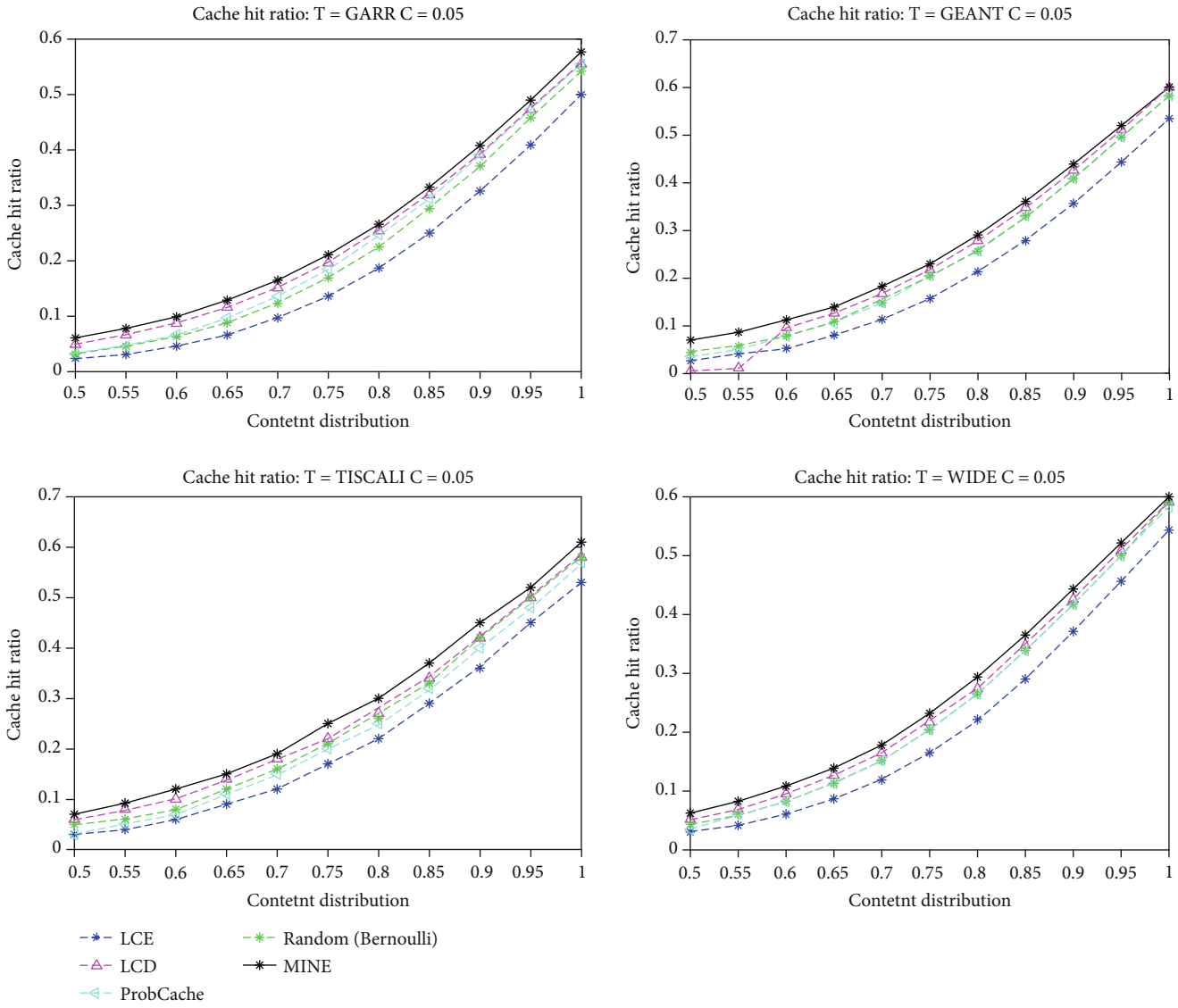

Figure 6: $C = 0.002$. The relationship between cache hit rate and content concentration.

(1) Prefilter queue length

$L(k) = L \times 2^{6-k}$, L is the length of the cache queue. The scheme used in the prefilter queue length in the simulation is an exponential function, which can effectively improve the hit rate.

(2) Prefilter queue number K maximum KMAX

$$\text{KMAX} = 5. \tag{3}$$

(3) Filter count

The maximum number of filtering for the prefilter queue is 1.

The parameters for the Icarus simulation platform are set as Table 3. Since the initial state of the cache queue is empty, the cache replacement will not occur until the cache queue is filled, so it will affect the calculation of the hit rate. In order to eliminate this part of the impact, Icarus supports the warm-up function. The previous part does not collect data, and the data is collected for the hit rate after the number of requested packets exceeds the warm-up value, and the warm-up value can be freely set by the user.

4.2. Simulation Scheme. Icarus provides several topology data sets in Topology Zoo [28]. This simulation is tested in GEANT, GARR, TISCALI, and WIDE four topologies, with DLRU-K cache replacement strategy, FIFO cache replacement strategy, LRU cache replacement strategy, and RAND cache replacement strategy.

4.2.1. GEANT Topology Simulation Scheme. GEANT is a pan-European data network for research and education

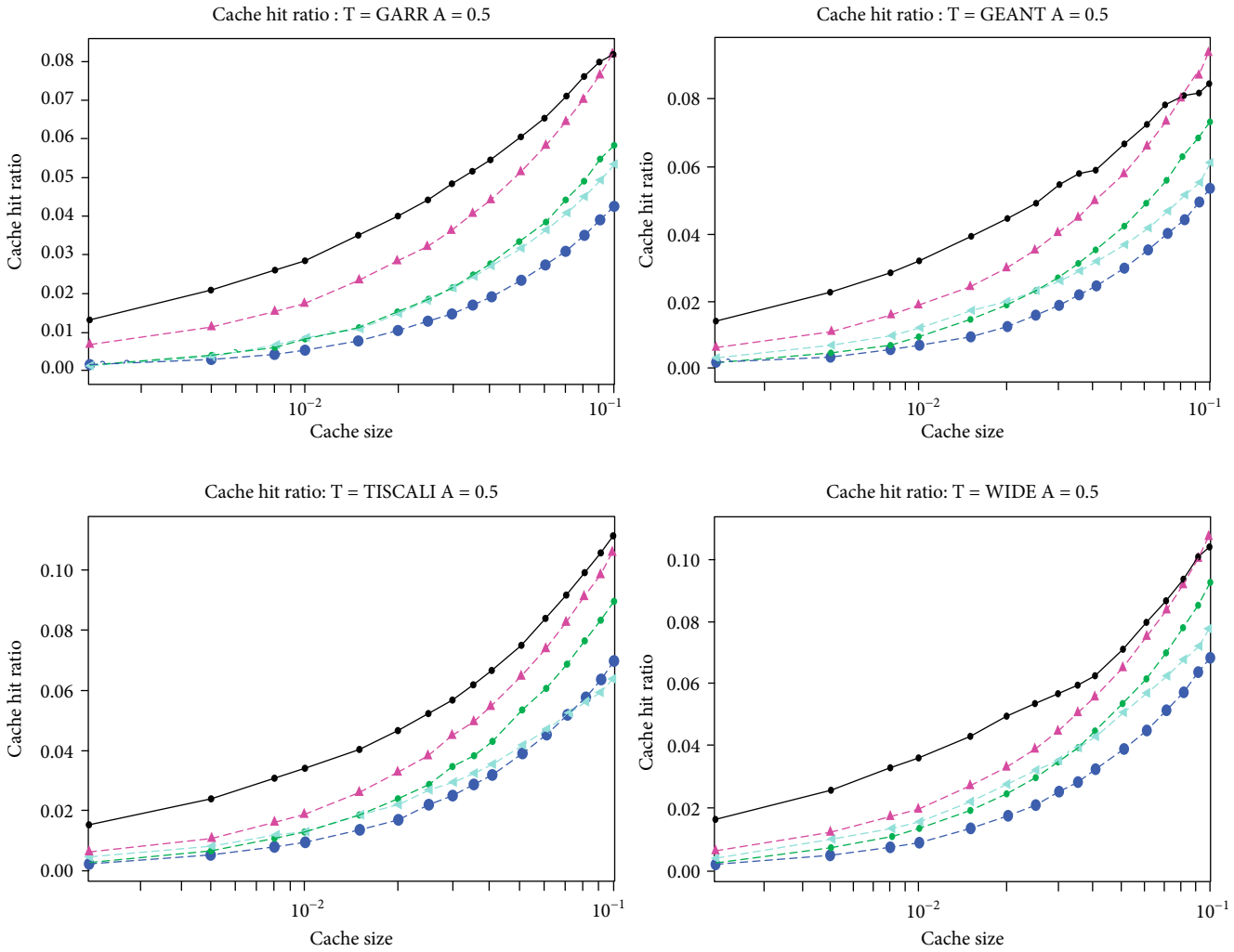

Figure 7: Relationship between cache hit ratio and cache capacity when $\alpha = 0.5$.

groups that connects research and education networks across Europe. In addition to providing high-bandwidth links in Europe, GEANT can also serve as a new technology test platform.

4.2.2. GARR Topology Simulation Scheme. GARR is a national computer network prepared for universities and research institutions in Italy. Its main goal is to design and manage a high-performance network infrastructure for the Italian academic and scientific communities. In fact, GARR has always been an integral part of the European research network GEANT, which shares GEANT with other European NRENs (national research and education network (NREN)). The GAR network topology provided by Icarus comes from Topology Zoo, and the version is GARR 201201.

4.2.3. WIDE Topology Simulation Scheme. Widely Integrated Distributed Environment (WIDE), an Internet project cofounded by Keio University, Tokyo Institute of Technology, and the University of Tokyo, is the backbone of the Japanese Internet and Japan's first Internet infrastructure.

4.2.4. TISCALI Topology Simulation Scheme. Since the real network topology of the real Internet service provider cannot be accessed by the research community, the Rocketfuel is used to map the nodes of the network. The TISCALI topology tested in this paper is mapped by Rocketfuel, and there are 44 content source nodes. There are 160 router nodes and 36 consumer nodes.

4.3. Simulation Result Analysis. The simulation results are mainly analyzed for the two performance indicators presented, the cache hit ratio and path scaling ratio under different network topologies. Cache hit ratio measures the response of cache nodes to interest packets in the network, which can intuitively reflect the efficiency of network cache deployment strategies. The path scaling ratio intuitively reflects the relative position of the cache deployment node on the path. The experimental results are as follows.

Figure 3 shows four cylindrical comparisons of the cache hit rates with different Zipf distribution parameters alpha and different cache sizes C. When the Zipf distribution parameters alpha = 0.5 and $C = 0.002$, the cache hit rates in the four network topologies have been significantly improved, which is higher than the experimental data of

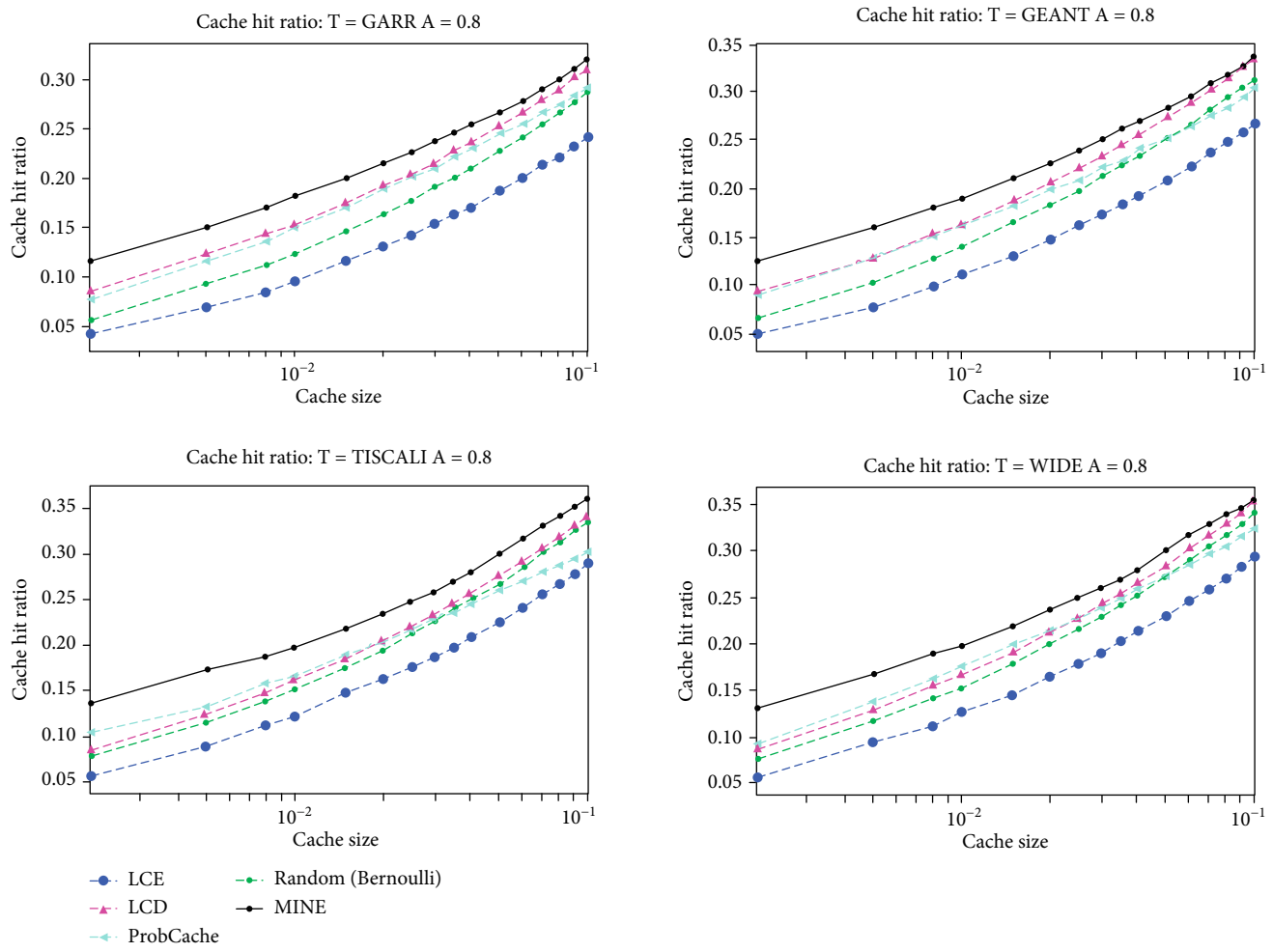

Figure 8: $\alpha = 0.8$, the relationship between cache hit rate and cache capacity.

the control group, and the fluctuation range of the hit rates is also smaller than the other four cache deployment strategies. The proposed caching strategy performs best among the five caching strategies, LCD is second only to the caching strategy in this paper, and the performance is relatively stable. However, the performance of ProbCache and Random in different network topologies is not stable and has certain volatility.

Compared with Figure 4, the path scaling ratio of the proposed cache strategy is also slightly lower than that of the other four control groups. By comparing the two groups of Figure 4, it can be found that the smaller the alpha and C, the more obvious the improvement of the cache hit rate is, but the reduction of path scaling ratio is not obvious. On the contrary, with the increase of alpha and C, the gap of path scaling ratio increases. The smaller the path scaling ratio is, the smaller the average request length is. Reducing the average request length not only brings faster response speed but also means better releasing the pressure of the core router so that interest packages can be responded at the edge of the network. For the forwarding process of each interest packet, even if the average number of hops is reduced by one hop, for the large number of interest packets existing in the network, it also means that the great load of the router is reduced, and the throughput of the router is increased.

The next part will mainly analyze the impact of different content popularity and cache size on the cache hit rate.

4.3.1. The Impact of Content Popularity on Cache Hit Rate. When $C = 0.002$ and $C = 0.05$, the break-line diagrams of cache hit rate are shown in Figures 5 and 6. The two break-line diagrams show that the cache strategy proposed in this paper is between alpha = 0.5 and alpha = 1.0. The cache hit rate in the four topologies is higher than that of other cache strategies. In GEANT and WIDE network topologies, when alpha reaches 1.0, the gap tends to decrease. With the increase of alpha value, the hit rate of all caching strategies increases, which indicates that the hit rate of caching increases with the increase of content concentration.

In the line graph, the cache hit rate of the five cache policies varies with the content popularity. The LCE cache deployment strategy has always been the worst performer. The hit ratio of LCD and ProbCache is closest to the cache strategy proposed in this paper. Even in the case of Figure 6, there is an almost equal situation. The caching strategy proposed in this paper gradually disappears when $\alpha = 1.0$.

4.3.2. The Effect of Cache Size on Cache Hit Ratio. The cache hit ratios when $\alpha = 0.5$ and $\alpha = 0.8$ are shown in Figures 7 and 8.

Through the two sets of broken-line graphs of Figures 7 and 8, it can be concluded that the cache hit rate of the four network topologies proposed in this paper is higher than that of other cache strategies when $C = 0.002$ to $C = 0.07$. However, when $C = 0.08$ and $C = 0.09$, the cache hit rate is almost equal to that of LCD strategy, and even when $a = 0.5$ and $C = 0.09$, the cache jitter occurs, which is surpassed by LCD strategy. The parameters of the scheme are not suitable for GEANT networks where C is greater than 0.08. According to other broken-line graphs, the cache hit rate of all caching strategies increases with the increase of caching, and the caching strategy awareness proposed in this paper is also getting smaller and smaller. According to the above analysis, the cache deployment strategy based on node status and location, combined with the dynamic LRU-K cache replacement strategy proposed in this paper, can improve the cache hit rate. The more decentralized the content and the smaller the cache, the more significant the enhancement effect.

5. Conclusions

ICN cache deployment strategy is divided into the noncooperative cache, implicit cooperative cache, and explicit cooperative cache. From the perspective of global cache hit rate and average request length, explicit cooperative caching generally has the best performance, but explicit cooperative caching requires a lot of communication information to determine the cached nodes, which will increase the network load. Therefore, this paper makes a thorough analysis and comparison of implicit cooperative caching. According to the advantages and disadvantages of other implicit cooperative caching, combined with the dynamic LRU-K cache replacement algorithm, a deployment strategy based on the status and location of caching nodes is proposed. According to the network location of caching nodes, the number of prefiltered queues, and the number of prefiltered times, the status is evaluated comprehensively, and the state value is put into interest. In the packet, the best state cache node is selected from the request path and cache in it when the packet returns. Finally, the deployment strategy based on the location and state of the cache node can reduce the duplication of the network cache space content and effectively improve the cache hit rates that are proved by the comparative simulation of the Icarus simulation environment.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant Nos. 61771153, 61831007, and 61971154) and the Fundamental Research Funds for the Central Universities (Grant No. 3072020CF0601).

References

[1] W. X. Liu, J. Li, J. Cai et al., "Cod: caching on demand in information-centric networking," *Telecommunication Systems*, vol. 69, no. 3, pp. 303–319, 2018.

[2] Y. B. Sun, Y. Zhang, and H. L. Zhang, "Survey of research on information-centric networking architecture," *Acta Electronica Sinica*, vol. 44, no. 8, 2016.

[3] V. Jacobson, D. K. Smetters, and J. D. Thornton, "Networking named content," in *Paper presented at the Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009.

[4] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," *Computers & Security.*, vol. 94, p. 101863, 2020.

[5] L. Cai, J. K. Wang, and X. W. Wang, "Path cost and node cost based caching strategy for information-centric network," *Journal of Chinese Computer Systems*, 2017.

[6] Y. Ding, Q. Zheng, and G. Chen, "Cooperative caching for ICN based on heat and cache replacement rate of node," *Computer Engineering*, 2018.

[7] L. M. Huang, Y. Guan, and X. G. Zhang, "On-path collaborative in-network caching for information-centric networks," in *Paper presented at the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017.

[8] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city," *Future Generation Computer Systems.*, vol. 107, pp. 433–442, 2020.

[9] G. Q. Zhang, X. H. Wang, and Q. Gao, "A hybrid ICN cache coordination scheme based on role division between cache nodes," in *Paper presented at the 2015 IEEE Global Communications Conference (GLOBECOM)*, 2015.

[10] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 2, pp. 737–750, 2018.

[11] 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. 32754–32768, 2018.

[12] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking//ICN," *ACM*, pp. 27–32, 2013.

[13] S. Wang, J. Bi, J. Wu, and A. V. Vasilakos, "CPHR: in-network caching for information-centric networking with partitioning and hash-routing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2742–2755, 2015.

[14] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical web caches," in *Paper presented at the IEEE International Conference on Performance, Computing, and Communications*, 2004.

[15] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine learning techniques," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2021.

[16] V. Pacifici and G. Dán, "Content-peering dynamics of autonomous caches in a content-centric network," in *Paper presented at the 2013 Proceedings IEEE INFOCOM*, 2013.

[17] H. Wu, J. Li, T. Pan, and B. Liu, "A novel caching scheme for the backbone of named data networking," in *Paper presented at the 2013 IEEE International Conference on Communications (ICC)*, 2013.

[18] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *Paper presented at the Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, 2013.

[19] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of Internet of Things," *IEEE Internet of Things Journal.*, vol. 7, no. 6, pp. 4682–4696, 2020.

[20] C. Bernardini, T. Silverston, and O. Festor, "A comparison of caching strategies for content centric networking," in *Paper presented at the 2015 IEEE Global Communications Conference (GLOBECOM)*, 2015.

[21] G. Q. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: a survey," *Computer Networks*, vol. 57, no. 16, pp. 3128–3141, 2013.

[22] M. Shafiq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, "Data mining and machine learning methods for sustainable smart cities traffic classification: a survey," *Sustainable Cities and Society*, vol. 60, p. 102177, 2020.

[23] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Paper presented at the Proceedings of the second edition of the ICN workshop on Information-centric networking*, 2012.

[24] J. Qiu, Y. Chai, and Z. Tian, "Automatic concept extraction based on semantic graphs from big data in smart city," in *IEEE Transactions on Computational Social Systems*, vol. 7no. 1, pp. 225–233, 2020.

[25] Y. Wang, Z. Tian, Y. Sun, X. Du, and N. Guizani, "LocJury: an IBN-based location privacy preserving scheme for IoCV," *IEEE Transactions on Intelligent Transportation Systems.*, vol. 22, 2020.

[26] A. S. Gill, L. D'Acunto, and K. Trichias, "Bidcache: auction-based in-network caching in ICN," in *Paper presented at the 2016 IEEE Globecom Workshops (GC Wkshps)*, 2016.

[27] L. Saino, I. Psaras, and G. Pavlou, *Icarus: A Caching Simulator for Information Centric Networking (ICN)*, Paper presented at the SimuTools, 2014.

[28] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.