PB-NCC: A Popularity-Based Caching Strategy with Number-of-Copies Control in Information-Centric Networks

Yuanhang Li, Jinlin Wang and Rui Han

Abstract: The Information-Centric Network (ICN), designed for efficient content acquisition and distribution, is a promising candidate architecture for the future Internet. In-network caching in ICN makes it possible to reuse contents and the Name Resolution System (NRS) makes cached contents better serve users. In this paper, we focused on the ICN caching scenario equipped with an NRS, which records the positions of contents cached in ICN. We propose a Popularity-based caching strategy with Number-of-Copies Control (PB-NCC) in this paper. PB-NCC is proposed to solve the problems of unreasonable content distribution and frequent cache replacement in traditional caching strategies in ICN. We examine PB-NCC with a large number of experiments in different topologies and workloads. The simulation results reveal that PB-NCC can improve the cache hit ratio by at least 8.85% and reduce the server load by at least 11.34% compared with other on-path caching strategies, meanwhile maintaining a low network latency.

Keywords: information centric network; caching strategy; popularity-based; name resolution

1. Introduction

With the explosive growth of the scale of Internet users and the network traffic, the Internet has grown into a bloated giant. According to the Cisco Annual Internet Report (2018–2023) [1], VR and HD video will take up most of the network traffic. In this context, the main goal of the Internet has gradually shifted from the point-to-point connection to the acquisition and distribution of contents. To meet this goal, the Information-Centric Network (ICN) has been considered as an important way to improve the existing network. By decoupling contents from the address and enabling in-network caching, ICN can better support some features such as mobility, multicast, and multi-hosting. With in-network caching, the ICN caching routers (or caching nodes) cache contents and provide content services, resulting in a lower latency in acquiring content and a smaller link traffic load on servers.

In recent years, various ICN architectures have been proposed, such as CCN [2], DONA [3], COMET [4], and PURSUIT [5]. ICN architectures can be divided into two categories when it comes to content discovery mechanisms accordingly. One category is routing-by-name, which couples content routing with the name of contents. A typical representative is CCN, which maintains a PIT (Pending Information Table) for request information and an FIB (Forwarding Information Base) table for content forwarding information. This mechanism requires ICN routers to handle mass storage and content information updates. The second category is routing-by-resolution, which utilizes a dedicated device named Name Resolution System (NRS). Basically, the NRS is responsible for recording the
mapping of the cached content and its caching location. In this way, users can get the NA (Network Address) of the content from the NRS. For example, DONA deploys at least one logical NRS in each autonomous system. The routing-by-resolution mechanism makes routers concentrate on routing and caching, thus alleviating their computing pressure. Correspondingly, the NRS needs to handle a large number of content name queries and updates. According to [6], NRS currently supports larger than $10^{15}$ content indexes. Another problem with the routing-by-resolution mechanism is the communication overhead with the NRS. To face this, some research focused on providing deterministic delay resolution services in the NRS [7].

As the core function of ICN, in-network caching has received a lot of attention. Different from Content Delivery Network (CDN), caching in ICN works at the network layer. Furthermore, ICN requires the caching router to be able to cache contents during their transmission and requires ensuring the wire-speed forwarding function, which makes it difficult for ICN to adopt a centralized content placement strategy. For simplicity, we use the term “caching strategies” instead for distributed content placement strategies in this paper. At present, a large number of caching strategies have been proposed to meet different optimization goals or application scenarios, such as Leave Copy Everywhere (LCE), Leave Copy Down (LCD [8]), Cache Less For More (CL4M [9]), and PROB [10]. However, the majority of the caching strategies are based on the routing-by-name mechanism, and only a few come from the routing-by-resolution mechanism. According to the information provided by the NRS, users can find the nearby content replica easily, and the routing-by-resolution mechanism can increase the hit ratio of the in-network cache and further reduce the server load. Therefore, how to design an appropriate caching strategy in the scenario with an NRS is worth further study.

Furthermore, two intractable problems limit the performance of traditional ICN caching strategies. One is improper distribution of contents. Aforementioned strategies do not consider information about cached contents when caching a new content, which makes it difficult for caching routers to persistently cache popular contents, resulting in low cache performance. Another problem is traditional strategies may lead to frequent cache replacement, since they do not restrict the content placement process. This will make the caching router spend more computing resources for caching operations, affecting the quality of packet forwarding and content service, as well as increasing energy consumption.

In this paper, our motivation is to design an efficient caching strategy to face the above problems in the ICN scenario equipped with an NRS. Considering that the NRS can retrieve the NAs of content replicas, any caching node has the potential to serve contents to all users. Based on this, our goal is to design a caching strategy to promote the cache hit ratio of caching nodes, as well as a lower latency in content acquisition. For the ICN caching scenario equipped with an NRS, we design a lightweight caching strategy taking into account the information from the NRS. The main contributions of this paper are as follows:

- We analyze the ICN caching scenario with an NRS and build a network model using the cache hit ratio as the optimization target. Based on the model, different caching strategies are compared and evaluated.
- We propose a popularity-based caching strategy with two new features in an ICN caching scenario. One feature is placing the content copy at the on-path caching router with the lowest cache popularity of contents, to gradually push popular contents to the cache close to users. Another is to use number-of-copies control, which is an implicit way to allocate cached resources. These two features allow for a better content distribution in the cache and reduce frequent cache replacement. To the best of our knowledge, our strategy is the first to control the number of content copies. This strategy can reduce the redundancy of content copies and promote the network performance.
- We compare the proposed caching strategy with other classic caching strategies in aspects including different topologies and workloads. Our caching strategy shows
obvious advantages in terms of the cache hit ratio and server load, meanwhile keeping the user delay of getting contents at a low level.

The rest of this paper is organized as follows: at first, we conclude the related work in Section 2. In addition, Section 3 depicts the ICN caching scenario equipped with an NRS as well as the problem formulation. Later, Section 4 describes our proposed caching strategy and the experimental design and results are displayed in Section 5. Section 6 concludes the paper and analyzes some possible future studies.

2. Related Works

Depending on the location of placing contents, caching strategies can be divided into two categories: on-path and off-path caching strategies. In on-path strategies, only the information along the content delivery path can be used for caching decisions. In addition, the dynamic environment will highly affect the on-path caching strategies, so these strategies are usually lightweight or heuristic with low computing and communication overhead [11]. LCE is the default on-path caching strategy in ICN and places a content replica on each on-path node. LCE is a simple strategy for pushing contents close to users, at the cost of bringing cache redundancy. In LCD [8], a new content replica is cached by only one hop downstream from the hit position of requests. CL4M [9] further considers attributes of node location and chooses the downstream node with the largest betweenness centrality for content placement. Similarly, PROB [10] makes decisions based on the cache space of downstream nodes so that contents will be cached at the edge of the network with a higher probability. EDGE caching [12] caches all contents at the edge of the network to reduce content transfer latency. The performance of EDGE caching is strongly influenced by user distribution, and it is challenging to avoid frequent content replacement of edge nodes. Ref. [13] proposed a Popularity-Based Collaborative Caching Algorithm in CCN to place popular content close to users. The algorithm requires the caching node to record the number of requests for each content that passes through this node. Since the content set is usually large, this algorithm will take up valuable cache space. Our previous work [14] proposed an on-path cache scheme for ICN hierarchical topologies. This scheme makes use of both content information and topology information and specifies an expected number of copies for each content. A shortcoming of this scheme is that it requires global information from content and topologies. In addition, the scheme needs to be modified to apply to arbitrary topologies.

Compared with on-path strategies, the off-path strategies usually require frequent information interaction between nodes, or a central processor is required to gather information of all nodes. An example is Hash Routing [15], a typical off-path strategy, which maps contents to specific nodes for caching. Hash Routing only caches one replica for each content, and there is limited space for improvement in network performance. Depending on where popular contents should be cached in the network, Ref. [16] proposed three off-path caching strategies. These strategies were evaluated only in a tree topology, and the implementation in arbitrary topology needs further adjustment. Ref. [17] proposed a multi-hop neighborhood collaborative caching strategy to make full use of the cache information of adjacent nodes. This strategy uses attenuated Bloom filters to record content information from adjacent nodes, but it also brings problems like a false-positive rate in Bloom filters and the frequent updates of cache state about adjacent nodes. Ref. [18] proposed a proactive caching placement algorithm for arbitrary topology, considering the information from content requests and user distribution. The algorithm reduces the hop count for content acquisition and improves the level of load balancing with the help of an ICN manager. The ICN manager is responsible for collecting content information and requesting information to make caching decisions, which needs powerful computing ability. However, the algorithm is limited when the content set is huge and dynamically changing.

At present, most proposed caching strategies are based on the routing-by-name mechanism, while only a few caching strategies take advantage of the NRS. Ref. [19] focused on optimizing the indexes of the NRS, and concluded that caching the most popular content
in the NRS can bring the most benefit, but there is no further consideration about how different caching strategies would affect the performance of NRS. Ref. [20] compared the performance of LCE, LCD, and EDGE caching under different request routing mechanisms. The results showed that LCE has no obvious advantage over EDGE caching, and applying the routing-by-resolution mechanism will bring more benefits. Ref. [21] proposed a caching scheme for edge resolution, to reduce the pressure of maintaining the cache state in NRS. Since cached contents in locations outside of edge nodes are not indexed, this scheme still has the potential for improvement.

As discussed above, under the routing-by-name mechanism, the performance improvement from on-path caching strategies is limited, while off-path caching strategies result in more communication overhead or require enough global information. As for the strategies under the routing-by-resolution mechanism, most of them only consider changes in the way requests are routed. The NRS gathers the information of requests as well as the content position, and this information has the potential to improve network performance.

3. Model Statement

In this section, we establish an ICN caching scenario equipped with an NRS and set up the problem formulation. Table 1 represents all the notations used in this paper.

Table 1. Summary of the notations.

| Name | Comment |
|------|---------|
| $V$  | Set of ICN caching nodes |
| $E$  | Set of links between nodes |
| $E_{i,j}$ | The link between node $i$ and node $j$ |
| $C$  | Set of contents |
| $s_k$ | Size of content $k$, $k \in C$ |
| $c_i$ | Cache space equipped in node $i$ |
| $\lambda_i^{in}$ | Inflow request rate in node $i$ |
| $\lambda_i^{out}$ | Outflow request rate in node $i$ |
| $p_{i,k}$ | Inflow request probability of content $k$ in node $i$ |
| $h_i$ | Cache hit ratio of node $i$ |
| $x_{i,k}$ | Boolean variable indicates whether node $i$ caches content $k$ |
| $h_i^{total}$ | Total cache hit ratio |
| $r_i$ | Total number of served request counts in node $i$ |
| $r_{i,k}$ | Count of served requests of content $k$ in node $i$ |
| $n_{base}$ | Base number of copies for the topology |

3.1. Scenario Description

We use $G = (V, E)$ to represent an arbitrary topology, where $V$ is the set of ICN caching nodes and $E$ is the set of links between nodes. For node $i$, $c_i$ is its cache size, and $E_{i,j}$ denotes the link between node $i$ and its neighbor node $j$. The ICN caching scenario with an NRS is depicted in Figure 1. Besides the aforementioned ICN caching nodes, the scenario has an NRS recording the mapping between the content name and the NAs, one or more source servers storing all contents persistently, and multiple users who continuously send requests for contents. The NRS is responsible for receiving registration messages for the content. In addition, the NRS will handle name resolution messages and respond to them with a list of the NAs of the content. This list is used by the user to further request the content. We use $C$ to denote the set of contents and for content $k \in C$, and $S_k$ represents the size of content $k$.

According to the routing-by-resolution mechanism in ICN, the content will be registered to the NRS before being served. In the example of Figure 1, when User1 wants to download the content, he or she needs to send a name resolution message for the corresponding content to the NRS, and the NRS returns an NA list of the content to User1. User1 gets the NA of the Source Server and sends a request to the Server. The Server returns the content to User1, meanwhile the on-path nodes R1 and R3 cache the content and register
the content to the NRS. User2 retrieves the content following the same process. User2 first sends a name resolution message and the NRS returns a list containing the NAs of the Servers, R1 and R3. Then, User2 chooses to get the content from R1 and sends a request to R1. Finally, R1 serves the request and returns the content to User2.

For users to get contents using the NA list, there are two simple ways: applying Nearest Replica Routing (NRR [20]) or applying random NA selection. In the first way, the user always sends a request to the nearest copy. However, in the second way, the user selects an NA at random from the NA list to send the request. We apply NRR in this paper, as the routing-by-resolution mechanism commonly sets NRR as the routing strategy.

Figure 1. An ICN caching scenario with an NRS.

3.2. Problem Formulation

Since the NRS enables all cached replicas in the network to provide services to any user, it is important for caching nodes to cache contents properly. The cache hit ratio represents the ability of the cache to serve requests, so it is usually used as an indicator to evaluate caching strategies. In general, a caching strategy with a higher cache hit ratio also results in lower server load and network latency. In this paper, we consider the cache hit ratio as the optimization goal. For a caching node $i \in V$, $\lambda^i_{in}$ denotes the inflow request rate and $\lambda^i_{out}$ denotes the outflow request rate. We use $p_{ijk}$ to denote the inflow request probability of content $k$ in node $i$ for $k \in C$, and we can easily find that $\sum_{k \in C} p_{ijk} = 1$. $x_{ijk}$ represents a boolean variable indicating whether node $i$ caches content $k$. $x_{ijk}$ is 1 if content $k$ is cached in node $i$ and 0, otherwise. When contents cached in the node $i$ serve the corresponding requests, these requests would no longer be forwarded. Thus, the relationship between $\lambda^i_{in}$ and $\lambda^i_{out}$ can be further expressed as $\lambda^i_{out} = \lambda^i_{in} \sum_{k \in C} (p_{ijk}(1-x_{ijk}))$. When $\lambda^i_{in}$ equals 0 the cache hit ratio of node $i$ (denoted as $h_i$) is obviously 0; when $\lambda^i_{in}$ gets bigger, $h_i$ can be estimated as $h_i = \frac{\lambda^i_{in} - \lambda^i_{out}}{\lambda^i_{in}} = 1 - \sum_{k \in C} (p_{ijk}(1-x_{ijk})) = \sum_{k \in C} p_{ijk}x_{ijk}$.

According to the above analysis, the cache hit ratio maximization problem can be expressed by the following equation:

Maximize:

$$h_{total} = \sum_{i \in V} h_i = \sum_{i \in V} \sum_{k \in C} p_{ijk}x_{ijk}$$ (1)
subject to:

\[ \sum_{k \in C} s_k x_{i,k} \leq c_i, i \in V \quad (2) \]

where \( s_k \) is the size of content \( k \) and \( c_i \) is the size of cache equipped in node \( i \). In Equation (2), the total space occupied by cached contents from node \( i \) cannot exceed its original cache size. From Equation (1), it is easy to find that more popular contents bring a higher cache hit ratio. To promote the cache hit ratio, an intuitive way is to cache popular contents and replace unpopular contents. Therefore, we need to choose popular contents when making content placement decisions.

As for the on-path caching strategy, a key question is how many copies need to be cached along the path. To improve the total cache hit ratio of the network, it is reasonable to place at most one copy at a time. When the popularity of the transferred content is low, it is not worth evicting popular content to cache it; on the contrary, for a particular delivery path, if the downstream node caches the content, requests for the content would not be forwarded upstream. Thus, placing more than one copy would waste the cache space of upstream nodes.

In addition, the cache hit ratio of the network is impacted by the request hit location. If the request of a popular content is hit on a Source Server, replacing unpopular content on a caching node with the popular content will directly increase the cache hit ratio. If the request is hit on a caching node, caching the popular content will not increase the total cache hit ratio, as requests for the content are just hit at different caching nodes. However, from the user’s perspective, getting content from closer caching nodes reduces more transmission latency. Therefore, the idea of caching popular contents at caching nodes closer to users has been used in many caching strategies, such as in Ref. [13]. Based on this, we aim to design a caching strategy that makes the caching nodes cache more popular contents.

Another useful condition is that the NRS can provide information of contents, such as the number of copies in the network (the number of NAs mapped to the content). Our previous work [14] has demonstrated that controlling the number of copies can significantly improve the network performance, since it can reduce cache redundancy and filter out unpopular contents, as well as reduce the probability of cache replacement occurring. Limiting the number of content copies can make more contents cached on the network. We consider further controlling the number of copies in ICN scenarios with the NRS to improve the performance of caching strategies.

4. Strategy Description

In this section, the brief design of the popularity-based caching strategy in the ICN caching scenario with an NRS is proposed. In the strategy, the on-path node that caches the least popular content is chosen to cache the transmitted content. In addition, the strategy controls the number of copies of all contents using the information from the NRS. We first illuminate two main parts of the strategy: popularity estimation and Number-of-Copies Control. Then, we show the expansion of the ICN packet header to support the strategy. After that, we show the strategy in detail in the ICN caching scenario with an NRS, including the processing of request packets and data packets.

4.1. Popularity Estimation

From the analysis in Section 3, caching nodes can gain a higher cache hit ratio when caching contents with a higher inflow request rate. We consider that the content popularity varies from caching nodes because the network position and the user distribution may change the inflow request rate of contents. There has been some research focused on popularity estimation [22,23]. However, the process of popularity estimation should be lightweight and simple due to the requirement of line-speed forwarding in ICN.

In caching node \( i \), we record the count of served requests of content \( k \) as \( r_{i,k} \) and the count of served requests as \( r_i \). Based on these, we can simply estimate \( p_{i,k} \) as \( r_{i,k} / r_i \). Further-
more, each node is responsible for recording the minimum popularity and corresponding content within the cache. The process of popularity estimation is triggered by the request packet and is depicted in Algorithm 1. If a caching node has free cache space, its minimum popularity is set as 0.

Algorithm 1: Popularity Estimation

1. caching node $i$ receives a request for content $k$;
2. if the destination NA of the request is node $i$ then
3.     $r_{i,k} = r_{i,k} + 1, r_i = r_i + 1$;
4.     if content $k$ is the content with minimum popularity then
5.         find the new content $k'$ with minimum popularity;
6.         record content $k'$ and the minimum popularity;
7.     end
8. end

4.2. Number-of-Copies Control

Controlling the number of copies can reduce redundancy and improve content diversity, especially when the content set is huge. From the analysis in Section 3, caching nodes tend to cache popular contents to promote the cache hit ratio. However, from the conclusion of our previous work [14], contents cached downstream will reduce the inflow request rate of the same contents cached upstream in ICN hierarchical topologies. In arbitrary topologies, this problem still exists. That is to say, as the number of copies increases, the marginal benefit of placing new replicas decreases. Thus, controlling the number of copies can help nodes determine whether to cache the transferred contents.

However, getting the expected number of copies of contents usually requires global information, such as the cache location of the content copies. When the cache location of the content copy has changed (such as when the content copy is evicted from a caching node), the expected number of these content copies also requires to be recalculated, which incurs heavy computational overhead. Therefore, we expect to use a lightweight Number-of-Copies Control approach. Through analysis and trade-offs, we considered two easily accessible factors that might affect the expected number of content copies. The first factor is the number of caching nodes in ICN, which reflects the size of the network and the upper limit of the number of content copies. The distance between users grows with the size of the network, to better serve the users utilizing caching, content copies need to be placed as close to the user as possible. For this reason, we speculate that larger networks will result in larger expected numbers of content copies. The second factor is the popularity of contents. It is intuitive to think that popular contents will be requested by more users, so having more copies for them will bring more network benefits.

Based on the above analysis, we specify a base number of copies (denoted as $n_{\text{base}}$) for the topology and use $\text{int}(n_{\text{base}} p_{i,k})$ to represent the expected number of copies of content $k$ predicted by node $i$. The base number $n_{\text{base}}$ is positively correlated with the topology size. As the user will get the NA list of content $k$ when sending a name resolution message to the NRS, we can compare the current number of copies and the expected number of copies of content $k$. If the current number of copies is smaller, we can perform caching operations while transferring content $k$. This comparison only needs to be done at the node that serves the request (this node is called a hit node) and will not increase the load of on-path nodes.

4.3. Popularity-Based Caching Strategy with Number-of-Copies Control

Our goal is to maximize the network cache hit ratio, as well as keeping the content transfer latency low. To avoid redundant content, we design the caching strategy as choosing at most one on-path caching node for content placement. The cache location is selected as the on-path node that carries the content of minimum popularity. Furthermore, the hit node will compare the current number of copies and the expected number of copies,
To decide whether to cache this content. We define the proposed strategy as PB-NCC for short.

To implement PB-NCC, we expand the Cache-Control field in the transport layer header of the ICN request packet and the data packet, as depicted in Figure 2. The Cache-Control field is made up of the Cache flag, the Popularity field, and the Copy Number field. The Cache flag is used to quickly filter out data packets for cache operation. If the Cache flag is 1, on-path nodes will further judge whether the caching conditions are met. Otherwise, the data packets will be forwarded to the user directly. The Popularity field in request packets is used to find the minimum popularity from on-path nodes, while those in data packets are used to find the eligible nodes for cache operation. The Copy Number field is used to record the current number of copies in the topology in request packets. The hit node will use this information to control the number of copies of contents.

PB-NCC is implemented by constructing and processing ICN request packets and data packets. The request packet is constructed by the user and shown in Algorithm 2. With the NA list returned from the NRS, the user gets the current number of copies and fills it into the Copy Number field of the request packet. The Popularity field in the request packet is initialized to the maximum, to find the minimum popularity of on-path nodes.

**Algorithm 2: Request Packet Construction**

1. The user sends a name resolution message for content $k$ and receives the NA list from the NRS;
2. set Cache flag to 0, set Popularity field to the maximum;
3. set Copy Number field to the current number of copies;
4. forward the request packet;

PB-NCC is implemented by constructing and processing ICN request packets and data packets. The request packet is constructed by the user and shown in Algorithm 2. With the NA list returned from the NRS, the user gets the current number of copies and fills it into the Copy Number field of the request packet. The Popularity field in the request packet is initialized to the maximum, to find the minimum popularity of on-path nodes.

The request packet forwarding process is shown in Algorithm 3. When a caching node $i$ receives a request packet, it will first check out whether the destination NA is itself. If yes, then node $i$ processes the popularity estimation and constructs and returns the data packets. Otherwise, node $i$ compares the value of the Popularity field in the request packet with the minimum popularity recorded itself and updates the Popularity field with the smaller value before forwarding the request packet.

The data packet is constructed in the hit node, as shown in Algorithm 4. If the hit node is a Source Server, the Number-of-Copies Control will not be triggered, for the reason that request hit at caching nodes other than at Source Servers will bring higher cache hit ratio and lower latency. Thus, in this case, the Cache flag is set to 1 and the Popularity field is set equal to that from the request packet for content $k$. If the hit node $i$ is a caching node, this node first needs to compare the Copy Number field and $\text{int}(n_{\text{base}P_{ik}})$. If the Copy Number field is smaller, meaning that more replicas of content $k$ can be cached in the topology. Thus, the Cache flag of data packets is set to 1, and the Popularity field is set
equal to that from the request packet for content $k$. If the Copy Number field is larger, all values in the Cache-Control field are set to 0, and on-path nodes will not cache content $k$.

**Algorithm 3: Request Packet Forwarding Process**

1. Caching node $i$ receives a request packet for content $k$;
2. if the destination NA in the request packet is node $i$ then
   3. update the popularity of content $k$ and the minimum popularity;
   4. construct and forward data packets to the user;
else
   6. if the minimum popularity of node $i$ is less than the Popularity field then
      7. update Popularity field as the minimum popularity of node $i$;
   8. end
   9. forward the request packet;
10. end

**Algorithm 4: Data Packet Construction**

1. hit node $i$ processes a request for content $k$;
2. if node $i$ belongs to Source Servers then
   3. set Cache flag to 1;
   4. set Popularity field equal to that from the request packet;
else if Copy Number field is smaller than $\text{int}(n_{\text{base}} p_{i,k})$ then
   6. set Cache flag to 1;
   7. set Popularity field equal to that from the request packet;
else
   9. set all the values in the Cache Control field to 0;
10. end
11. forward the data packet;

The data packet forwarding process is shown in Algorithm 5. The on-path node $i$ first checks out the Cache flag of the data packet. If the Cache flag is 1, node $i$ continues to compare the Popularity field and the minimum popularity of node $i$. If the Popularity field is not smaller than the minimum popularity, then node $i$ is chosen for caching the content, and the Cache flag is set to 0. Otherwise, the node $i$ will directly forward the packet without caching or modifying it.

**Algorithm 5: Data Packet Forwarding Process**

1. caching node $i$ receives a data packet of content $k$;
2. if Cache flag is 1 then
   3. if the minimum popularity of node $i$ is not larger than Popularity field then
      4. cache the data packet;
      5. set Cache flag to 0;
   6. end
7. end
8. forward the data packet;

4.4. Discussion on the Proposed Strategy

The efficiency of the proposed strategy. The proposed strategy chooses the caching node that caches the least popular content for content placement, which can obviously increase the average popularity of the contents cached by the selected nodes. Average popularity refers to the average value of the popularity of different contents in the cache. Furthermore, the average popularity of cached contents on the transfer path has also been
improved. As the least popular content is more likely to be replaced, the proposed strategy can increase the average popularity of cached contents on the path more significantly. Choosing to cache popular contents and using number-of-copies control can also reduce the number of cache placement, which in turn reduces the frequency of cache replacement. This saves the computing resources of the cache router and reduces power consumption. In addition, the proposed strategy is decoupled from content replacement policies, meaning that, when using PB-NCC, we can select different content replacement policies according to actual scenarios.

**Path cooperation or regional cooperation.** The proposed strategy belongs to path cooperation. However, there may be caching nodes that cache less popular contents outside the path. For regional cooperation, there are usually two kinds of approaches. The first way is to adopt neighbor collaboration mechanisms to get the state of the neighbor nodes. The second way is to adopt centralized decision-making in order to search for appropriate caching nodes. These approaches are often used in off-path caching strategies, as discussed in Section 2, which inevitably brings communication overhead to the network. In addition, as the unpopular contents are frequently replaced in the cache, caching nodes also need to frequently exchange the content information with each other, and this takes up more computing resources for caching nodes. Based on these, we adopt path cooperation rather than regional cooperation.

**The diffusion rate of popular contents.** Pushing popular contents to the caching nodes that are close to users will increase the cache hit ratio, as well as reduce the transfer latency of contents. However, on the other hand, the frequent content replacement will affect the serviceability of edge nodes. In the proposed strategy, content placement happens when the popularity of the transmitting content is higher than the least popular content cached on the path. This criterion prevents unpopular content from contaminating the cache near the edge, and the popular content can be progressively pushed to the edge. On the other hand, the content replica selection mechanism such as NRR in the ICN scenario with the NRS can further ensure the diffusion rate of popular contents, for the reason that users will access the nearest content replica. This means that popular content rarely travels over long paths.

5. Experiment Simulation

We further estimate PB-NCC in this section. We firstly introduce the experimental setup. Then, we analyze PB-NCC in the tree topology and find a proper base number of copies for the topology. Next, we compare the proposed strategy with the state of the art in aspects including content distribution and replacement in the cache. Finally, we compare PB-NCC with other strategies in different topologies and different workloads.

5.1. Experimental Setup

We set up the experiment simulation based on Icarus [24], a Python-based discrete-event simulator for evaluating ICN caching performance. We implement the routing-by-resolution mechanism for caching strategies, as well as realizing the popularity-based caching strategy in Icarus. We compare PB-NCC with four widely used on-path caching strategies: LCE, LCD, CL4M, and PROB. Table 2 shows the basic parameters for the simulation. To evaluate the performance of the caching strategies in different topologies, we further select four real-world topologies: EBONE (87 nodes), TELSTRA (108 nodes), ABOVEnET (141 nodes), and SPRINTLINK (315 nodes) from Rocketfuel [25]. Two types of workloads are used for simulation. The Zipf distribution workload has a content set with $10^5$ contents, with popularity defined by the Zipf parameter $\alpha$. This content size was analyzed in [26] and can be considered appropriate to simulate a realistic scenario. Based on previous research on web content [27,28], it is reasonable for $\alpha$ to range from about 0.4 to 1.0. We set $10^5$ requests for system warm-up and another $10^5$ requests for performance measurement. The YouTube workload [29] has a content set with $5.81 \times 10^5$ contents, and the total request number is set to $1.46 \times 10^6$, from which we use the first $4.6 \times 10^5$ requests.
for system warm-up and the remaining for performance measurement. All requests follow
the Poisson distribution, and 10 requests are generated by each user per second. All
caching nodes are allocated the same cache space and use the LRU [30] policy for content
replacement. A unified content replacement policy is more advantageous to compare the
performance of the different caching strategies.

For each scenario, we run the experiment 10 times and average the results. Each
experiment measures three indicators: cache hit ratio, network average latency, and average
server load. Cache hit ratio refers to the proportion of content requests served by caching
nodes, and network average latency refers to the average latency for users to get contents.
Average server load represents the average load on the servers during the experiment. In
addition, we will evaluate the performance of caching strategies under different cache size
ratios, which represent the ratio of cache space to the total size of the content set.

Table 2. Experiment parameters.

| Parameters                        | Value                                           |
|-----------------------------------|-------------------------------------------------|
| Topology                          | Tree topology/ISP like topologies               |
| Traffic workload                  | Zipf distribution/YouTube video traffic         |
| Range of Zipf parameter $\alpha$  | $0.4–1.0$                                       |
| Size of the content set           | $10^5/5.81 \times 10^5$                        |
| Request number                    | $2 \times 10^7/1.46 \times 10^6$               |
| Request distribution              | Poisson distribution                           |
| Request rate                      | 10 req/s                                        |
| Cache allocation                  | Uniform                                         |
| Content replacement               | LRU                                             |
| Number of experiments             | 10                                              |

5.2. Performance on the Tree Topology

We set up a ternary tree topology with a depth of 4 (40 caching nodes in total), and
users connect the leaf nodes evenly. One Source Server and an NRS are connected to the
root node. The link delay between the Source Server and the root node is set to 50 ms and
the delay of other links is set to 2 ms.

5.2.1. Finding the Appropriate $n_{base}$

As discussed in Section 4.2, we use $n_{base}$ to specify a base number of copies for the
topology. Firstly, we need to find the appropriate $n_{base}$. The base number of copies $n_{base}$
is supposed to reflect the properties of the topology. For any content, the upper limit of
its number of copies equals the number of caching nodes in the topology (in this case,
every caching node will cache one copy). Thus, an intuitive assumption is that $n_{base}$ is
proportional to the number of nodes. As the number of caching nodes is 40, we set $n_{base}$ to
$(10, 40, 160)$ and unlimited (without controlling the number of copies).

Figure 3 shows the performance of PB-NCC with different $n_{base}$ when the Zipf param-
eter $\alpha$ changes, while Figure 4 shows the performance when the cache size ratio changes.
From the simulation results, we can find that applying the Number of Copies Control
(NCC) will bring a higher cache hit ratio and lower server load. As for the network latency,
applying NCC will reduce it in most parameters. Larger $n_{base}$ results in a poor cache
hit ratio but could reduce network latency, for the reason that, as the number of copies
increases, content copies are more likely to be closer to users. In general, setting $n_{base}$ equal
to the number of caching nodes will get near the highest cache hit ratio and the lowest
server load, as well as the lower latency in most cases. Thus, we set $n_{base}$ equal to the
number of caching nodes in the following experiments.
5.2.2. Comparison in Content Distribution and Cache Replacement

As mentioned above, the ideal content distribution in the cache is that contents with higher popularity are cached closer to users, which makes more requests hit at the edge routers. To verify the results of content distribution, we calculate the cache hit ratio of caching nodes at the same depth of the tree topology and calculate the average. Figure 5 shows the average cache hit ratio of different cache strategies when node depths changes. The root node has a depth of 1, and the leaf nodes have a depth of 4. From Figure 5, we can find out, in LCE, nodes of different depths have a similar cache hit ratio. This is because LCE does not take into account the location of the node or content information. LCD and CLAM place content more at the core nodes, resulting in the low cache hit ratio at the edge nodes. In PROB, more requests are hit at nodes with larger depth, which improves the network performance. By contrast, in PB-NCC, most requests are served by leaf nodes, which would significantly reduce the content transfer latency. The average cache hit ratio grows when the depth grows in PB-NCC. This also reflects the hierarchical content distribution in the network: popular contents are cached at the edge, and less popular contents are cached near the core.
Next, we count the number of cache replacement that occurs under different caching strategies. Frequent cache replacement will affect the forwarding and content service capabilities of the cache router. Figure 6 shows the average number of cache replacement per node in different caching strategies. PB-NCC has the smallest number of cache replacement because both popularity-based process and number-of-copies control reduce unnecessary content placement operations. PROB also has a relatively low number of cache replacement because it caches contents selectively. The number of cache replacement in LCE, LCD, and CL4M is larger since they place more copies during content transmission, leading to more energy consumption.

5.2.3. Comparison in Zipf Distribution Workload

We further compare the proposed caching strategy with the state of the art in Zipf distribution workload. Figure 7 shows the comparison when $\alpha$ changes, while Figure 8 shows the comparison when the cache size ratio changes. PB-NCC gets the highest cache hit ratio, the lowest server load, and most of the time the lowest latency. By local popularity estimation, PB-NCC keeps popularity contents in the cache. PROB shows the advantage of low latency when the cache size ratio is small, for the reason that PROB makes caching decisions based on the cache space available downstream. When cache space is small, PROB makes better use of limited cache space. As the cache size ratio increases, this advantage wears off. In tree topologies, both LCD and CL4M cache contents in the next hop of the hit node, so they share the same simulation results.
Figure 7. Comparison in the tree topology when the Zipf parameter $\alpha$ changes. (a) the cache hit ratio under different Zipf parameter $\alpha$; (b) the average latency under different Zipf parameter $\alpha$; (c) the server load under different Zipf parameter $\alpha$.

Figure 8. Comparison in the tree topology when the cache size ratio changes. (a) the cache hit ratio under a different cache size ratio; (b) the average latency under a different cache size ratio; (c) the server load under a different cache size ratio.

5.2.4. Comparison in YouTube Traffic Workload

In YouTube video traffic workload, we set the cache size ratio to (0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0) and represent the simulation results with the log of the x-coordinate. As shown in Figure 9, PB-NCC still keeps its advantage over different evaluation indicators. LCE has a slight advantage in latency when the cache size ratio is small. Results show that PB-NCC works well under a real-world traffic workload. Content popularity in YouTube traffic has a time locality, which means that content popularity changes over time. PROB performs poorly as it cannot always guarantee to keep popular content. LCD and CL4M are also hard to cache contents effectively in time-varying traffic due to their slow diffusion rate of contents.

5.3. Performance on the ISP-like Topologies

In the ISP-like topologies, $n_{base}$ is set equal to the number of caching nodes. Unlike tree topologies, to be more realistic, we set multiple Source Servers in ISP-like topologies, and users are distributed throughout the network. The NRS is connected to the ICN caching node who has the maximum betweenness centrality. The link delay between each Source Server and the directly connected ICN router is set to 50 ms and the delay of other links is set to 2 ms.

Figure 10 shows the performance of different caching strategies in ISP-like topologies under the Zipf distribution workload when $\alpha$ changes. In different topologies, PB-NCC keeps its advantages over cache hit ratio, latency, and server load. PB-NCC improves the cache hit ratio by 8.85% at least and reduces the server load by 11.34% at least compared with other strategies. PROB and LCE perform similarly. LCD and CL4M perform poorly in
all topologies, due to their preferences that tend to cache content at the core rather than at the edges.

Figure 9. Comparison in the tree topology under YouTube traffic workload. (a) the cache hit ratio under a different cache size ratio; (b) the average latency under a different cache size ratio; (c) the server load under a different cache size ratio.

Figure 10. Comparison in ISP-like topologies when the Zipf parameter $\alpha$ changes. (a) performance in EBONE topology under a different Zipf parameter $\alpha$; (b) performance in TELSTRA topology under a different Zipf parameter $\alpha$; (c) performance in ABOVENET topology under a different Zipf parameter $\alpha$; (d) performance in SPRINTLINK topology under a different Zipf parameter $\alpha$.

5.3.1. Comparison in Zipf Distribution Workload

Figure 11 shows the performance of different caching strategies in ISP-like topologies under Zipf distribution workload when the cache size ratio changes. PB-NCC keeps its advantages in EBONE, TELSTRA, and ABOVENET, while in SPRINTLINK, PROB has a
slight advantage when the cache size ratio is small. Due to the large scale of SPRINTLINK, each caching node is allocated less cache space, PROB makes better use of the limited space. However, PROB performs poorly when the cache size ratio increases, and the performance of PB-NCC is improved significantly with a larger cache size ratio.

Figure 11. Comparison in ISP-like topologies when the cache size ratio changes. (a) performance in EBONE topology under a different cache size ratio; (b) performance in TELSTRA topology under a different cache size ratio; (c) performance in ABOVENET topology under a different cache size ratio; (d) performance in SPRINTLINK topology under a different cache size ratio.

5.3.2. Comparison in YouTube Traffic Workload

Figure 12 shows the performance of different caching strategies under YouTube traffic workload. PB-NCC performs well over cache hit ratio and server load and performs the second-best over latency. PB-NCC independently predicts the content popularity in each node, which is closer to the actual situation and can better track the changes of local popularity. In terms of latency, LCE performs well under YouTube traffic workload. One reason may be that both the content set and the cache space are large, and LCE pushes content to the edge more effectively in this condition. PROB performs the worst, meaning that when content space and cache space are large, using the information of cache space to make caching decisions does not work well. In comparison, better performance can be achieved with simple LCE.
5.4. Discussion

From the above analysis, PB-NCC is well adapted to different scenarios. The popularity estimation process takes advantage of the content information at a low cost, while the Number-of-Copies Control effectively reduces cache redundancy. PB-NCC may be further improved by using a more complex and precise method to predict local content popularity. For example, we can adopt the adapted Exponential Weighted Moving Average (EWMA) method proposed in [31] to track changing content popularity. Bayesian inference is also a lightweight method for popularity estimation [32]. These methods can be incorporated into our strategy to further improve cache performance. Moreover, we only consider the NA list provided by the NRS to make caching decisions. As all register messages and name resolution messages are first aggregated to the NRS, the NRS can provide approximate global information. This information is useful to perceive content popularity and user distribution.

After further analysis of the comparative strategies, we find that, under a Zipf distribution workload, all strategies get better performance with a larger $\alpha$. When the total cache space is small, making caching decisions using the information of cache space is helpful, like PROB. When the total cache space is large, it is economic to cache content close to users, like LCE. These findings help further design caching strategies.

6. Conclusions

In this paper, we address the design of the caching strategy in the ICN scenario equipped with an NRS. Using the information provided by the NRS, we propose a
Popularity-based Number-of-Copies Control (PB-NCC) caching strategy. The proposed caching strategy makes use of the content popularity estimated by on-path nodes and the number of content replicas from the NRS. Compared with classical caching strategies, PB-NCC reaches the highest cache hit ratio and the lowest server load in most cases, at the same time keeping the latency at a low level. PB-NCC performs well in real-world topologies, as well as under the real-world traffic workload. In addition, we are surprised to find that LCE also shows good performance in most cases. When the information for designing caching strategies is limited, LCE is an economic choice. Our future direction focuses on the performance improvement in network delay of PB-NCC, as well as making better use of the NRS for caching decisions.

Author Contributions: Conceptualization, Y.L., J.W. and R.H.; methodology, Y.L., J.W. and R.H.; software, Y.H.; writing—original draft preparation, Y.L.; writing—review and editing Y.L., J.W. and R.H.; supervision, J.W.; project administration, R.H.; funding acquisition, J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to express our gratitude to Jinlin Wang, Rui Han, Xu Wang, Li Zeng, and Yi Liao for their meaningful support for this work.

Conflicts of Interest: The authors declare that there is no conflict of interests regarding the publication of this paper.

References
1. Cisco, U. Cisco Annual Internet Report (2018–2023) White Paper. 2020. Available Online https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/whitepaper-c11-741490.html (accessed on 26 March 2021).
2. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking named content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Orlando, FL, USA, 9–12 December 2009; pp. 1–12.
3. Koponen, T.; Chawla, M.; Chun, B.G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007; pp. 181–192.
4. García, G.; Beben, A.; Ramón, F.J.; Maeso, A.; Psaras, I.; Pavlou, G.; Wang, N.; Śliwiński, J.; Spirou, S.; Soursos, S.; et al. COMET: Content mediator architecture for content-aware networks. In Proceedings of the 2011 Future Network & Mobile Summit, Warsaw, Poland, 15–17 June 2011; pp. 1–8.
5. Trossen, D.; Parisis, G. Designing and realizing an information-centric internet. IEEE Commun. Mag. 2012, 50, 60–67. [CrossRef]
6. D’Ambrosio, M.; Dannerwitz, C.; Karl, H.; Vercellone, V. MDHT: A hierarchical name resolution service for information-centric networks. In Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking, Cambridge, MA, USA, 14–15 November 2011; pp. 7–12.
7. Liao, Y.; Sheng, Y.; Wang, J. A deterministic latency name resolution framework using network partitioning for 5G-ICN integration. Int. J. Innov. Comput. Inf. Control 2019, 15, 1865–1880.
8. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of LRU caches and its analysis. Perform. Eval. 2006, 63, 609–634. [CrossRef]
9. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache “less for more” in information-centric networks. In International Conference on Research in Networking; Springer: Berlin, Germany, 2012; pp. 27–40.
10. Psaras, I.; Chai, W.K.; Pavlou, G. Probabilistic in-network caching for information-centric networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking; ACM: New York, NY, USA, 2012; pp. 55–60.
11. Ioannou, A.; Weber, S. A survey of caching policies and forwarding mechanisms in information-centric networking. IEEE Commun. Surv. Tutor. 2016, 18, 2847–2886. [CrossRef]
12. Fayazbakhsh, S.K.; Lin, Y.; Tootoonchian, A.; Ghodsi, A.; Koponen, T.; Maggs, B.; Ng, K.; Sekar, V.; Shenker, S. Less pain, most of the gain: Incrementally deployable icn. ACM SIGCOMM Comput. Commun. Rev. 2013, 43, 147–158. [CrossRef]
13. Zhu, X.; Qi, W.; Wang, J.; Sheng, Y. A Popularity-Based Collaborative Caching Algorithm for Content-Centric Networking. J. Commun. 2016, 11, 886–895. [CrossRef]
14. Li, Y.; Wang, J.; Han, R. An On-Path Caching Scheme Based on the Expected Number of Copies in Information-Centric Networks. *Electronics* 2020, 9, 1705. [CrossRef]

15. Saino, L.; Psaras, I.; Pavlou, G. Hash-routing schemes for information centric networking. In *3rd ACM SIGCOMM Workshop on Information-Centric Networking*; ACM: New York, NY, USA, 2013; pp. 27–32.

16. Dräxler, M.; Karl, H. Efficiency of on-path and off-path caching strategies in information centric networks. In Proceedings of the 2012 IEEE International Conference on Green Computing and Communications, Besancon, France, 20–23 November 2012; pp. 581–587.

17. Saino, L.; Psaras, I.; Pavlou, G. Hash-routing schemes for information centric networking. In *3rd ACM SIGCOMM Workshop on Information-Centric Networking*; ACM: New York, NY, USA, 2013; pp. 27–32.

18. Dräxler, M.; Karl, H. Efficiency of on-path and off-path caching strategies in information centric networks. In Proceedings of the 2012 IEEE International Conference on Green Computing and Communications, Besancon, France, 20–23 November 2012; pp. 581–587.

19. Shafi, S.; Feng, C.; Zhang, T.; Loo, J. Proactive caching placement for arbitrary topology with multi-hop forwarding in ICN. *IEEE Access* 2019, 7, 149117–149131. [CrossRef]

20. Chen, X.; Wang, L.; Ott, J.; Kangasharju, J.; Sathiaselvan, A.; Crowcroft, J. On content indexing for off-path caching in information-centric networks. In Proceedings of the 3rd ACM Conference on Information-Centric Networking, Tokyo, Japan, 26–28 September 2016; pp. 102–111.

21. Zhang, F.; Zhang, Y.; Raychaudhuri, D. Edge caching and nearest replica routing in information-centric networking. In Proceedings of the 2016 IEEE 37th Sarnoff Symposium, Newark, NJ, USA, 19–21 September 2016; pp. 181–186.

22. Zhu, X.; Wang, J.; Wang, L. An Information-Centric Networking Caching Scheme for Edge Resolution. *J. Netw. New Media* 2019, Volume 8, pp. 31–36.

23. Wang, J.; Han, R.; Liu, X. An On-Path Caching Scheme Based on the Expected Number of Copies in Information-Centric Networks. *Electronics* 2020, 9, 1705. [CrossRef]

24. Saino, L.; Psaras, I.; Pavlou, G. Icarus: A caching simulator for information centric networking. *Comput. Netw.* 2016, 103, 67–83. [CrossRef]