electronics

MDPI

Article

An On-Path Caching Scheme Based on the Expected Number of Copies in Information-Centric Networks

Yuanhang Li [1,2], Jinlin Wang [1,2] and Rui Han [1,2,*]

[1] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, No. 21, North Fourth Ring Road, Haidian District, Beijing 100190, China; liyh@dsp.ac.cn (Y.L.); wangjl@dsp.ac.cn (J.W.)

[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, China

* Correspondence: hanr@dsp.ac.cn; Tel.: +86-1-381-107-7380

Received: 18 September 2020; Accepted: 14 October 2020; Published: 17 October 2020

Abstract: The Information-Centric Network (ICN) is one of the most influential future network architectures and in-network caching in ICN brings some helpful features, such as low latency and mobility support. How to allocate cache capacity and place content properly will greatly influence the performance of ICN. This paper focuses on the cache allocation problem and content placement problem under the given cache space budget. Firstly, a lightweight allocation method utilizing information of both topology and content popularity is proposed, to allocate cache space and get the expected number of copies of popular content. The expected number of copies represents the number of content copies placed in the topology. Then, an on-path caching scheme based on the expected number of copies is proposed to handle the content placement problem. In the cache allocation scenario, the lightweight allocation method performs better than other baseline methods. In the content placement scenario, Leave Copy Down (LCD) based on the expected number of copies performs the second-best and is very close to Optimal Content Placement (OCP).

Keywords: ICN; cache allocation; expected number of copies; content placement

1. Introduction

The Information-Centric Network (ICN) has gained widespread attention in recent years. Different from host-to-host communication, ICN focuses on the content object itself, instead of the location. In ICN, content becomes the first-class citizen in the whole network and it is decoupled from physical addressing. By naming content at the network layer, ICN supports in-network caching, multicast and mobility mechanisms, thus making delivery of content to the users more efficiently and timelier [1]. Besides, ICN can be realized as a network slice in 5G architecture, to support multiple scenarios with low latency and high reliability [2].

In-network caching is the key factor of ICN. To realize in-network caching, routers (called as nodes in this paper) are equipped with cache and these routers construct caching networks. Content providers manage these routers and hire cache space to keep content replicas, to reduce the traffic load on the origin server that provides content [3]. Users can get data from caching nodes rather than the remote source server. In this way, the network latency will be decreased significantly.

For a content provider, despite the potential benefits of in-network caching, it also needs to consider the cost of hiring the cache resource. Because of the financial constraints and competition from other content providers, it is not practical to hire cache without limit. The financial constraint can also be converted to a total capacity constraint for simplicity. To promote network performance, cache distribution also needs to be considered and the sum of cache capacity for all nodes is usually limited.

If a cache node is close to users and with a high user access rate, then this node tends to be allocated more cache space. So to maximize the network performance with a given capacity constraint, how to allocate cache capacity to each node needs to be considered and we call this problem *the cache allocation problem*. In References [4,5], cache capacity was recommended allocated to nodes close to users rather than core nodes. Manifold learning was used in Reference [6] to calculate the importance of the node, while graph-related centralities of nodes were used for cache allocation in Reference [7]. These methods improve network performance to some extent, while [7] concluded that gain brought by heterogeneous cache capacity is very limited. The cache allocation problem is a complex optimization problem and factors such as network topology, user distribution, caching strategies or content popularity will have a potential impact on network performance. From a content-based perspective, the cache allocation problem can be formulated as a *content placement problem*: under the same cache space budget, placing content into the network to obtain the largest benefit. References [8,9] considered the Zipf distribution of web content [10] and network performance is further improved.

On one side, ubiquitous caching in ICN brings tangible benefits to both users and remote servers. On the other side, redundant replica and low content diversity limit the benefits of in-network caching. To improve this deficiency, we introduce the concept of the Expected Number of Copies (ENC)—each content cached on an ICN node has an ENC, to control the content number of copies cached on this node and downstream nodes. For popular content, a larger ENC allows more copies cached in the topology. For non-popular content, a smaller ENC allows fewer copies or no copy to be cached. How to calculate and apply the ENC is a question worth studying. Besides, ICN nodes need to make independent caching decisions or they can perform limited collaborative operations. So an appropriate caching mechanism is required to ensure network performance during data transfer.

In this paper, we focus on the cache allocation problem and the content placement problem under a given cache space budget, with the optimization goal: to minimize the average network hop count from data transfer. Particularly, this paper concerns about how each ICN node makes cache decisions during data transfer. Firstly, we propose a two-step lightweight method called LAM (Lightweight Allocation Method) to solve the cache allocation problem, as well as get the ENC of different content. LAM is suitable for the cache allocation scenario of all kinds of caching networks, such as ICN, CDN (Content Distribution Network) and cloud networks. Then, an on-path caching scheme based on ENC is proposed to handle the content placement problem in ICN. To the best of our knowledge, we are the first to calculate the optimal number of content copies in ICN and apply it for caching decisions. The main contributions of this paper are as follows:

- We build an ICN hierarchical network model using the reduction of network hop count as the optimization target. We formulate the cache allocation problem as the network benefit maximization problem in the model we establish.
- We propose a lightweight allocation method in ICN, which takes network information and content information into account. The method distributes the total cache budget across all nodes and places content to nodes, meanwhile maximizing network benefit. This method calculates the ENC of different content, to guide content placement. This method considers the newly placed copy will affect the benefit from existing copies, which is more realistic.
- We propose an on-path caching scheme based on ENC. ENC is used to control the number of content copies, within the scope of a node and its downstream nodes. Each node manages the ENC of cached content and allocates an appropriate number of copies to downstream nodes. ENC reduces content redundancy and guarantees content diversity.
- In the cache allocation scenario, the lightweight allocation method shows obvious advantages and performs better than all other baseline methods. In the content placement scenario, Leave Copy Down (LCD) based on ENC performs the second-best and is very close to Optimal Content Placement (OCP [8]).

The rest of this paper is organized as follows: the related work is presented in Section 2. Section 3 displays a hierarchical ICN topology as well as the problem formulation. Section 4 describes the lightweight allocation method for generating ENC. Section 5 describes the on-path caching scheme based on ENC. Section 6 presents the experiment design and results. Section 7 gives a consideration of the improvement direction of our scheme. Section 8 concludes the paper and analyzes some possible future studies.

2. Related Works

In-network caching has become the main research in ICN and performance optimization for caching can be carried out in a wide range of dimensions, such as cache resource allocation, content placement strategies and cache space sharing [11]. Cache resource allocation is the precondition and foundation of in-network caching and content placement is the key to promote cache benefit. Numerous studies have focused on one or more of these dimensions.

2.1. Cache Resource Allocation

Cache resource allocation has been applied in multiple scenarios. In Reference [12], Chu et al. aimed to allocate cache resources among multiple content providers according to their utilities. Cache space was partitioned into slices with each slice being dedicated to one content provider. However, this paper only considered promoting the cache hit ratio and the network latency was not involved. The authors of Reference [13] focused on allocating cache resources to different applications for mobile edge computing. Cache resource was only from a single base station and the cache hit ratio was not involved. Furthermore, a collaborative cache allocation method was proposed in Reference [14] to handle the situation with multiple base stations but it brought in extra communication overhead.

Homogeneous cache resource allocation, which means that all the nodes are equipped with the same size of cache space, has been used by default in many studies. Huo et al. mainly focused on differentiated caching for different classes of content, for the sake of avoiding the waste of caching resources and enhancing the cache availability [15]. The authors of Reference [16] allocated cache space in terms of game theory and they used a sequential auction mechanism to maximize social welfare. These papers used homogeneous cache size, while heterogeneous cache allocation was not considered as an aspect of the optimization. Banerjee et al. used a greedy algorithm to maximize the network hit rate for solving the content placement problem [17], while the cache size of each node was a random value of a given interval.

Heterogeneous cache allocation schemes have been the main part of some studies such as [7,18–20]. The authors of Reference [7] studied graph-related centralities metrics and they noticed that the gain brought by the heterogeneous cache is very limited. Besides, they found that the degree centrality already proves to be a "sufficiently good" allocation criterion and this criterion tends to deploy more space in the "core" nodes. However, topologies used in Reference [7] contained less than 100 nodes. When topology size increases, the topology structure becomes more complex and users become more fragmented. So conclusions in Reference [7] do not apply to large topology size. In Reference [18], the authors mainly studied the cache allocation and request routing in ICN and they observed that keeping larger cache space at the edge is naturally suited for an incremental deployment path for ICN. While content information was not taken into account in Reference [18].

Apart from considering topology only, References [19,20] also took network traffic into account. In Reference [19] cache was allocated according to short-term parameters such as user behaviors and outgoing traffic as well as network topology. This method is suitable for networks where content changes dynamically but cannot allocate cache properly before gathering enough traffic information. Duan et al. proposed a fine-grained cache deployment scheme to allocate cache [20], while fine grain size should be chosen properly. Besides, References [8,9] took content popularity into consideration. They proposed optimal allocation methods and found that if content popularity is more uniformly distributed, more cache should be allocated to the "core" nodes. However, these two methods ignored

the fact that the placement of new copies will affect the benefit brought by existing copies, leading to a discrepancy between the computation and the real situation. Moreover, these optimal allocation methods have high computational complexity, for the reason that they need to traverse all nodes many times, as well as searching the whole content space to get the optimal solution. When topology size grows or the content set increases, it becomes harder to calculate the optimal solution.

2.2. Content Placement

Classical caching policies mainly focused on simple placement strategies. Leave Copy Everywhere (LCE) made content cached in all nodes along the path, resulting in a large number of redundant copies. In Leave Copy Down (LCD [21]), a new copy was cached one hop downstream from the location of the hit node. LCD reduced redundancy but slowed down the speed pushing new content to the edge. Compared with LCD, Move Copy Down (MCD [22]) moved the copy to one hop downstream from the hit node, to further reduce content redundancy. However, content copies from core nodes would be deleted frequently, resulting in a decrease of the cache hit ratio.

In "Edge Caching [23]", the content was only cached at the edge of the network and users could get content with low latency. But this strategy would quickly make the edge cache reach its limits, leading to frequent content replacement. The authors of Reference [24] proposed a strategy called "CL4M (Cache Less for More)" to place content in nodes with high graph-based centrality. A content-based centrality is introduced and used in content placement in Reference [25] and the results show that content-based centrality performs better than other graph-related centralities. However, content-based centrality is hard to calculate, as content replacement happens frequently.

Several studies focused on optimal content placement in cache networks [8,26]. The authors of Reference [26] considered constraints including link bandwidth, disk space and content popularity but content prefetching was required to realize the optimal solution. Besides, other measures were required to locate the nearest content copy. Azimdoost et al. proposed an online optimal placement algorithm, in which each node kept a constant solution space to the cache [8]. Content in the solution space would be cached permanently, while content out of the solution space would be replaced by LRU strategy. The solution space for each node should be calculated by a coordination entity and it was recorded to solve the optimal content placement problem. Using solution space is simple and effective, for the reason that it can effectively reduce content redundancy, meanwhile using few computational resources. The key step of this algorithm was to obtain the solution space but this step in Reference [8] still ignored that the benefit from existing copies will be affected by newly placed copies.

3. Problem Statement

In this section we establish a hierarchical network model, as well as setting up the problem formulation, using network hop count as the optimization target. Table 1 represents the main notations used in this paper.

Table 1. Summary of the notations.

Name	Comment
T_{budget}	Total cache budget
H_s	Hop count from the root node to the server
$H_{i,j}$	Reduced hop count for content i at node j
N_m	Number of middle nodes
N_l	Number of leaf nodes
p_i	Request probability of content i
S_i	Size of content i
M	Set of nodes in the topology
M_k	A set records the nodes to place the content copy when the number of copies is k
B_k	The benefit of placing the content copy when the number of content copies is k
C	Set of content
E_j	Cache space equipped in node j
C_j	Set of content cached in node j
$r_{i,j}$	Request arrival rate of content i in node j
Content_num_dict	A dictionary in which key is the content and value is the number of content copies in the topology

3.1. Model Description

We choose the three-layer architecture to set up our model, to comprehensively analyze how topology size affects network benefit [27]. It should be noted that the ICN hierarchical topology of any number of layers still applies to the model. Figure 1 shows a three-layer hierarchical network containing one root node in the first layer connecting to one server, N_m middle nodes (also denoted as aggregation nodes) in the second layer and N_l leaf nodes in the third layer. The server keeps all the content permanently and is far away from users. For simplicity, all middle nodes are connected to the root node and connected to at least one leaf node. On the other hand, each leaf node only connects to one middle node. All nodes can be equipped with cache space to cache content. If a request arrives at a node that contains the requested content, the requested data will be sent from the hit node rather than from the origin server. This three-layer network can be easily extended to a larger multilayer tree topology, such as doing k-core decomposition for networks in Reference [20]. We assume that all users are connected to the leaf nodes and their requests for content will arrive at leaf nodes first.

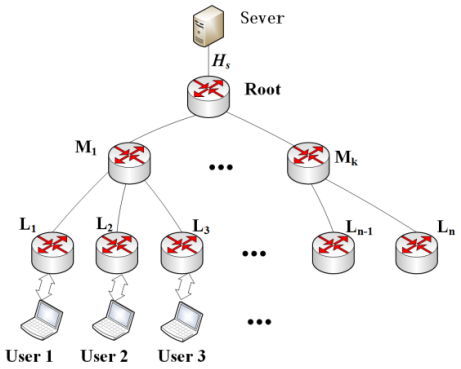


Figure 1. A three-level Cache Network System Model.

In this paper, to simplify the model, we make the following assumption:

All the leaf nodes are connected to the same number of users and the request frequency of the users is the same. Namely, all leaf nodes have the same request arrival rate.

Request routing is along the path to the origin server and the on-path cached content is capable to serve the request. Some measures such as applying the nearest replica routing strategy [23] or applying some specific request routing algorithms like [28] could also make use of content cache out of the path, while these methods will bring more traffic overhead. These measures are beyond the scope thus not discussed.

Nodes in the topology are represented by serial numbers, so as content. Furthermore, Content is named according to its popularity ranking.

3.2. Problem Formulation

The cache allocation problem and content placement problem share the same goal. Under the same cache space budget, they can be both solved by finding the optimal content copy distribution with maximal network benefit. We analyze these two problems in the scenario with one content provider, who owns a server storing all the content and plan to rent cache space under a given budget. First of all, a mechanism is required to compute the network benefit. In ICN, the cache benefit is usually measured by the reduction of the hop count to get the requested content [8,9]. Assuming a request for content i is sent to node j, which is the first encountered node with a cached copy of the content i. Compared with responding to the request at the server, the reduction of hop count equals the hop from node j to server, presented as $H_{i,j}$. Using p_i to present the content probability of content i, then the benefit, which is denoted as $B_{i,j}$, equals $H_{i,j}p_i$. For a node, responding to more requests will bring more benefits. Here we use $r_{i,j}$ to denote the number of requests asking for content i for node j will receive. Finally, we can calculate the benefit $B_{i,j}$ using Equation (1).

$$B_{i,j} = H_j r_{i,j} p_i. \tag{1}$$

It is worth noting that the benefit of a node will be affected by its downstream nodes. Suppose that there are only two copies of content i located at node M_1 and L_1 respectively in the topology of Figure 1 and all users request content at the same frequency. On this occasion, requests for content i from user 1 will be served in node L_1 and requests from user 2 and user 3 will be served in node M_1. So H_{i,M_1} is the hop count from node M_1 to server, while H_{i,L_1} is the hop count from node L_1 to M_1, i.e., one hop, for the reason that node M_1 will serve the request if the node L_1 does not cache content i. Besides, if a new copy of content i is placed to node L_2, the benefit from content i in node M_1 will decrease, according to the reduction of r_{i,M_1}.

In more general cases, the network benefit can be expressed using the following equation:

Maximize:

$$\sum_{j}^{M} \sum_{i}^{C_j} H_{i,j} r_{i,j} p_i \tag{2}$$

Subject to:

$$\sum_{i}^{C} p_i = 1 \tag{3}$$

$$\sum_{i}^{C_j} S_i \leq E_j, \ jM \tag{4}$$

$$\sum\nolimits_{j}^{M} \sum\nolimits_{i}^{C_j} S_i \leq T_{budget}, \tag{5}$$

where M is the set of nodes, C is the set of content, C_j is the set of content cached in node j, E_j is the cache space equipped in node j and $H_{i,j}$ represents the reduced hop count for content i at node j. Equation (4) indicates that the cache space allocated from node j cannot exceed its original cache size. In Equation (5), the sum of the cache capacity of all nodes cannot exceed the given total cache capacity T_{budget}.

3.3. Multiple Copy Placement of a Single Content

The benefit maximization problem is a special type of knapsack problems. Thus it cannot be solved in polynomial time. The benefit of content i at node j may change, as $H_{i,j}$ is affected by the

upstream nodes, while $r_{i,j}$ is affected by the downstream nodes. Current optimal allocation methods proposed in References [8,9] work in the case that the benefit from one content copy is unchanged, so cannot be directly used in our model.

Based on the analysis of network benefits, a new copy will only affect the network benefits of existing copies of the same content. So we can calculate total network benefits by calculating the network benefits of different content and sum them up. We firstly analyze a special case of a single content in the network and then conclude the general features from this example. The example is shown in Figure 2a, which consists of a server, a root node, two middle nodes and six leaf nodes. Content copies are placed step by step, aiming for the highest current network benefit. We use H_s to denote hop count from the root node to the server, r to denote request arrive rate at each leaf node and content popularity is set to 1.

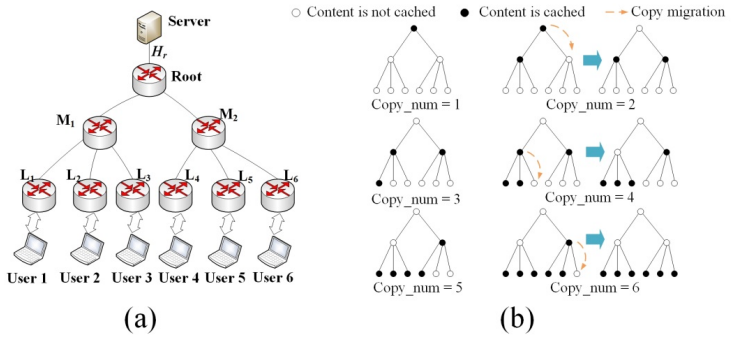


Figure 2. A case of placing copies of single content. (**a**) A three-level Cache Network System Model. (**b**) Multiple copy placement process of a single content.

To place the first content copy, the benefits of the root node, each middle node and each leaf node are $6rH_s$, $3r(H_s + 1)$ and $r(H_s + 2)$. As long as H_s is larger than one hop, placing content at the root node will bring maximal benefit. To place the second content copy, we calculate the nodes bringing the highest benefit are node M_1 and M_2. We choose M_1 to place the second copy but it is not the optimal placement solution when the number of content copies is 2. It is easy to find that if we move the copy from root node to M_2, the network benefit will increase $3r$ (3 access leaf nodes with the request arrival rate r and each request hit at M_2 will reduce one hop, from M_2 to root node). The similar situation happens when the number of copies equals 4 or 6. Continue placing copies, we can get the content placement process as shown in Figure 2b.

Two general features can be concluded from the example of Figure 2. Firstly, Content copy migration to child nodes may happen, to gain higher network benefit. Such migration only happens when a node caches the content copy but only one child node does not cache the copy. In this situation, migrating the copy to this child node will save one hop for some requests. Secondly, when the number of copies equals N_l, the network benefit becomes the maximum. At this point, the optimal decision is placing one copy on every leaf node, for the reason that leaf nodes are closest to clients. The hop count for every request reduces to the minimum and a larger placed number of copies will not bring more benefit but cost more space.

We have already explored the relationship between the number of copies and network benefits, meanwhile recording content distribution in the network. The benefit information will be used to maximize the current network benefit and content distribution information will be used to locate the content position. Based on them, we can place the proper content greedily to seek the maximal network benefit.

4. Lightweight Allocation Method

In this section, the brief design of the Lightweight Allocation Method (LAM) is proposed. LAM solves the cache allocation problem under the given cache capacity budget, as well as gives the

Expected Number of Copies (ENC) of popular content. We first illuminate the motivation to propose a lightweight allocation method. Then we show the specific steps of the algorithm in detail, as well as analyze the computational complexity of our method.

4.1. Motivation

As we have discussed before, heterogeneous cache allocation schemes have been proved efficient with large topology size. And the cache allocation problem requires a combination of many factors. Current lightweight allocation methods, such as graph-centralities-based methods, do not take advantage of the content information, resulting in poor performance. While most allocation methods using content information have high computational complexity. Besides, recent optimal allocation methods assume the network benefit of the content copy is fixed and our model is closer to the actual situation.

Based on the above discussion, we want to propose a lightweight allocation method, which considers both topology information and content information. As the problem raised in Section 3.2 is NP-hard, the greedy method can be applied to get the approximate optimal solution. Different from the optimal allocation methods [8,9], we only focus on popular content. As the request workload for different content obeys heavy-tailed distribution, only the content that brings the maximal benefit will be taken into account. by getting the ENC of popular content, we can get the near-optimal cache allocation solution and use ENC to guide content placement.

4.2. Method Description

LAM is based on the above analysis. The overview of LAM is shown in Figure 3 and the detailed step is presented in Algorithm 1. The input *Network_topology* contains the necessary topology information, T_{budget} is the total cache capacity to be allocated, M is the set of nodes and C is the set of content. The output $\{M_1, M_2, \ldots, M_{N_l}\}$ contains N_l sets, each of which records the content copy position and the subscript is the number of copies in the network. This output can locate the nodes to place the content copy. The output *Content_num_dict* records the relationship between content and its ENC in the topology. With these two output results, we can get the content placement solution, as well as the exact space to be allocated of each node.

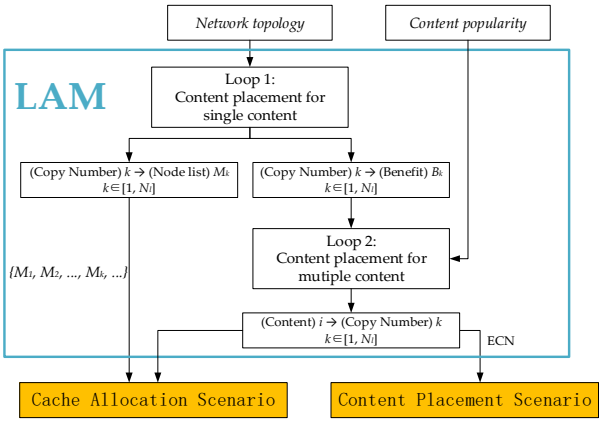


Figure 3. Overview of Lightweight Allocation Method (LAM).

The proposed LAM consists of two main loops. The first loop, from line 2 to 16 in Algorithm 1, computes the network benefit B_i, where i is the number of copies, as discussed in Section 3.3. The list $\{M_1, M_2, \ldots, M_{N_l}\}$ is also generated from this loop and M_k records nodes that have the copy when the number of copies is k. The upper bound of the number of copies is N_l, i.e., the size of leaf nodes. For the *k-th* copy ($k > 1$), $M - M_{k-1}$ is the set of nodes to choose for placement. We place a new copy based on the previous placing result and greedily choose the node that brings the maximal benefit to place

the copy. After updating M_k, in line 12 to 16, we need to check whether to move the copy from the upstream nodes to downstream, as discussed in Section 3.3. Then, we record the benefit as B_k, which will be used in the second main loop.

The second loop, from line 17 to 33 in Algorithm 1, is the placement process of multiple content. Firstly, we define *new_c* as the most popular content which has not been placed in the topology yet. Each time to place a content copy, we compute the benefit of *new_c* and benefits of content from C_{in}, which records current content that is already placed in the topology. For content i, its ENC in the topology is $num = Content\_num\_dict.get(i)$. When num is less than N_l, the benefit of placing a new copy of content i can be computed as $p_i B_{num+1}$. In line 22, we compare the benefit per unit size of different content, to avoid the situation where some content is profitable but takes up too much space. After finding the maximal benefit per unit size of a content copy, we update the ENC of the chosen content. This placing process ends when the deployed cache space T_{deploy} reaches T_{budget}.

The computational complexity of the first main loop is $O(N_l|M|) \leq O\left(|M|^2\right)$ and the second main loop is $O(T_{budget}\log(T_{budget}))$. So the overall complexity of our allocation method is $max(O\left(|M|^2\right), O(T_{budget}\log(T_{budget})))$. The computational complexity of optimal allocation method in Reference [20] is $max(O\left(|M|^3\right), O(T_{budget}\log(|C|)))$. Compared with the optimal allocation methods, the computational complexity of our method is relatively low. $|M|$ and $|C|$ are two main factors affecting the computational complexity. When T_{budget} is relatively small, $|M|$ becomes the major impact factor and the computational complexity ratio of our method to the optimal allocation method is $\frac{1}{|M|}$. When T_{budget} is relatively large, the computational complexity ratio is $log\frac{T_{budget}}{|C|}$. Content space size $|C|$ is much larger than the budget T_{budget}, so our method also gets a low complexity in this case.

Algorithm 1: LAM

Input: *Network_topology*, T_{budget}, M, C, P
Output: $\{M_1, M_2, \ldots, M_{N_l}\}$, *Content_num_dict*
1: **initialization:** $M_0 = \varnothing$, $C_{in} = \varnothing$, $Content\_num\_dict = \varnothing$, $T_{deploy} = 0$
2: **for** k from 1 to N_l do
3: $M_k = M_{k-1}$, $max\_benefit = 0$, $chosen\_node = -1$
4: **for** node j in $M - M_{k-1}$ do
5: compute $j\_benefit$ for placing copy at node j
6: **if** $j\_benefit > max\_benefit$
7: $chosen\_node = j$
8: $max\_benefit = j\_benefit$
9: **end if**
10: **end for**
11: $M_i.add(chosen\_node)$
12: **if** *chosen_node.parent* accords with moving criteria
13: update M_i and $max\_benefit$
14: **end if**
15: $B_k = max\_benefit$
16: **end for**
17: **while** $T_{deploy} < T_{budget}$ do
18: set *new_c* as the most popular content from $C - C_{in}$
19: compute $new\_benefit$ for placing *new_c*
20: find $max\_benefit$ for the content with maximal benefit in C_{in}
21: set *chosen_c* as the content with maximal benefit in C_{in}
22: **if** $\frac{new\_benefit}{S_{new\_c}} > \frac{max\_benefit}{S_{chosen\_c}}$ **then**
23: $max\_benefit = new\_benefit$
24: $chosen\_c = new\_c$
25: **end if**
26: $C_{in}.add(chosen\_c)$
27: **if** *chosen_content* **in** *Content_num_dict.keys*
28: $copy\_num = Content\_num\_dict.get(chosen\_c)$
29: $Content\_num\_dict.add(key = chosen\_c, value = copy\_num + 1)$
30: **else** $Content\_num\_dict.add(key = chosen\_c, value = 1)$
31: **end if**
32: $T_{deploy} += S_{chosen\_c}$
33: **end for**
34: return $\{M_1, M_2, \ldots, M_{N_l}\}$, *Content_num_dict*

5. ENC-Based On-Path Caching Scheme

Although LAM gives a near-optimal solution of content placement, for each content, the set recording the node position is required during the data transmission process. This information will consume additional bandwidth [8]. Besides, only using the optimal solution of content placement makes nodes lack autonomous decision-making abilities and the robustness and extensibility of the network will be limited.

To better fit ICN, we design an on-path caching scheme using ENC for content placement. Different on-path caching policies can be set, with the control of ENC of different content. Due to the trace-driven feature in ICN, user requests also potentially contain the location information, which helps to place content copies properly. Since the transmission path of data packets is from the core node to edge nodes, through the extended field of the data packet header, upstream nodes can indirectly manage the number of content copies from downstream nodes.

5.1. Scheme Description

The proposed on-path caching scheme uses ENC to control the number of content copies in the network. The origin server calculates the ENC of each content. When the content is transferred to the network, the first ICN node will cache the content and record its ENC. ENC represents the upper limit of the number of copies in a range, which includes the node that records this ENC and all its downstream nodes. With ENC, upstream nodes can pre-determine the number of content copies to be cached downstream. When an ICN request hits at an ICN node, this node will recalculate the ENC for downstream nodes. This ENC-based caching scheme applies to all on-path caching strategies.

An example of the caching scheme is shown in Figure 4. Suppose that the ENC of content i_1 and i_2 is 10 and 3 in turn, The server maintains the ENC of all content and informs the on-path nodes during the transmission of data packets. So when the root node caches content i_1 and i_2, it will record their ENC. Besides, the root node will recalculate the ENC of its child nodes. When M3 node caches content i_1 and i_2, the ENC of content i_1 and i_2 is allocated as 3 and 1. Similarly, L6 will only cache content i_1, as its upstream node M3 allocates ENC of content i_1 is 1 for L6 and ENC of content i_2 is 0.

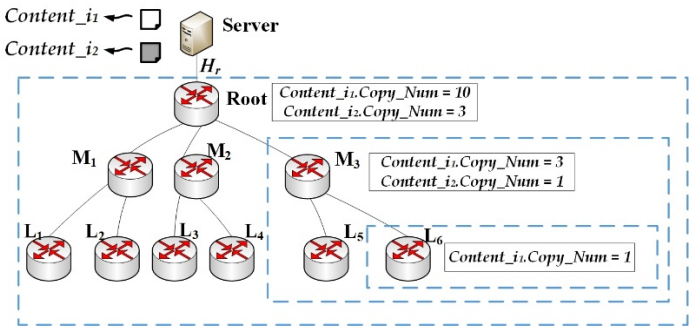


Figure 4. An example of the Expected Number of Copies (ENC)-based caching scheme.

The ENC can be carried in the header of ICN data packets. We extend the cache control field in the transport layer header of ICN data packets, shown in Figure 5. The cache control field includes three parts, the cache flag field, the cache strategy field and the copy number field. The cache flag field is used to quickly filter data that needs to be cached. If the cache flag is set to 0, all ICN nodes will forward the packet directly without caching it. This situation happens when the data packet is unpopular so on-path nodes will not cache it or the upstream node has cached it and modified the cache flag according to the cache strategy like LCD. If the cache flag is 1, the ICN nodes will cache the packet and further check the cache strategy field. The cache strategy field is used to instruct ICN nodes to execute different caching policies. The copy number field is used to control the number of content copies. When the ICN node decides to cache content, this node will record the value of the

copy number field and recalculate the number of copies to be cached downstream. Lastly, the node will update the copy number field and forward the packet downstream.

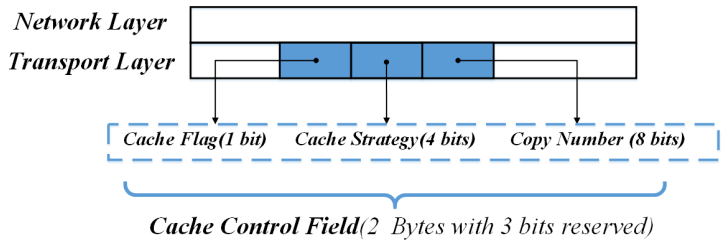


Figure 5. Cache control field design.

5.2. ICN Packet Forwarding Process

The ENC-based caching scheme is implemented by constructing and processing ICN data packets. The key step is to modify the copy number field in the data packet. This allows downstream nodes to know how many copies should be placed. To modify the copy number field, the simplest way is allocating ENC equally according to the router port number or the degree of nodes. The second way is allocating ENC according to the traffic weight of router ports. The third way is allocating ENC according to the cache space of downstream nodes and this way requires nodes to know the cache space information of downstream nodes.

We choose the traffic weight for the allocation of ENC. The traffic information is influenced by the user's activity degree, so it is more helpful compared with port number and cache space. When a node caches a content copy, the ENC of the content is also recorded. When a node plans to send content to the user, it will first find the out port to forward the content. Then, the node will get the traffic information of this out port and calculate the proportion of this out port traffic in the total traffic. Then, this node allocates the ENC according to the traffic proportion weight, to guide downstream nodes making caching decisions.

The request packet forwarding process is shown in Algorithm 2. When the request hits the cache at node j, it will set the cache control field. i_ECN is used to record the expected number of copies for content i. out_port denotes the physical port of node j to forward the content i and Tr_{out_port} denotes the traffic of out_port from node j. Tr_{total} denotes the total traffic from node j. When i_ECN is larger than 1, meaning that downstream nodes should cache more copies to get close to the expected number of copies. Each downstream node directly connected to node j will record the expected number of copies for content i as $\text{int}\left(i\_ECN * \frac{Tr_{out\_port}}{Tr_{total}} + 0.5\right)$ when caching content i.

Algorithm 2: Request Packet Forwarding Process

ICN node j gets a request packet for *content i*
if node j has *content i*
  read the i_ECN for *content i*, Tr_{out_port} and Tr_{total} from metadata
  if $i\_ECN > 1$
    set *cache flag* to 1
    set *cache strategy field* according to the caching strategy configuration
    set *copy number field* as $\text{int}\left(i\_ECN * \frac{Tr_{out\_port}}{Tr_{total}} + 0.5\right)$
  else
    set *cache flag* to 0
    set *copy number field* to 0
  end if
  construct and forward the data packet
else
  forward the request packet
end if

The data packet forwarding process is shown in Algorithm 3. When the cache flag is 0, on-path nodes will forward the packet to the destination without caching it or modifying the cache control field. When the cache flag is 1, the processing node will firstly check whether to modify the cache flag according to the caching strategy configuration. If the packet has been cached by the processing node, this node will directly forward it with the updated cache control field. Otherwise, the node will cache the packet before forwarding it.

Algorithm 3: Data Packet Forwarding Process

ICN node j gets a data packet belonging to *content* i
if *cache flag* $== 0$
    forward the data packet
else
    set *cache flag* according to the caching strategy configuration
    if node j has *content* i
        update i_ECN in node j with *copy number field*
        update *copy number field* as int$\left(i_ECN * \frac{Tr_{out_port}}{Tr_{total}} + 0.5\right)$
        forward the data packet
    else
        cache the data packet at node j
        record i_ECN in node j with *copy number field*
        update *copy number field* as int$\left(i_ECN * \frac{Tr_{out_port}}{Tr_{total}} + 0.5\right)$
        forward the data packet
    end if
end if

6. ICN Scenario Simulation

The lightweight allocation method for cache allocation and the on-path caching scheme based on ENC for content placement have been clarified. Next, we introduce the process of experiments in this section. We firstly introduce the experiment parameters. Then, we present experimental results analysis in the cache allocation scenario and content placement scenario.

6.1. Experimental Setup

Table 2 shows the basic parameters for our simulation. The experiments run on the Python-based ICN cache simulation platform. The server permanently stores 10^5 content with popularity defined by Zipf distribution. The Zipf distribution parameter α and content set size $|C|$ determine the popularity of the content i, using the following equation: $p_i = \frac{i^{-\alpha}}{\sum_{k=1}^{|C|} k^{-\alpha}}$. All users generate requests with the same probability. We generate 10^6 requests for the system to warm up and another 10^6 requests for data collection in each experiment. Each experiment runs 10 times and the average result is calculated. From the previous research on web content [8,29], the Zipf distribution parameter α ranges from about 0.4 to 1, so we set α based on this range. The sum of cache capacity from each layer is recorded, to observe how different experimental parameters affect cache distribution. The average hop count from all requests is measured during each experiment. Besides, we also measure the cache hit ratio, which can reflect the traffic load on the origin server.

Table 2. Experiment parameters.

Parameters	Value
Number of topology layer for cache allocation	3
Number of topology layer for content placement	5
Content space size	10^5
Requests number for system warm-up	10^6
Requests number for system data collection	10^6
Experiment run time for each scenario	10
Range of topology size	50–300
Range of total cache space	100–1100
Range of Zipf parameter α	0.4–1.0

In the cache allocation scenario, LCE (Leave Copy Everywhere) content placement strategy is used by default in the topology and the LRU cache replacement strategy [22] is taken for the cache eviction of each cache node. Three traditional cache allocation methods, including "Uniform","Degree Centrality [7]" (denoted as "Degree") and "Edge Caching [18]" (denoted as "Edge") are used to compare with LAM. "Uniform" distributes cache capacity evenly among all cache nodes. "Degree" allocates cache capacity according to the degree centrality of the node. In "Edge" only the leaf nodes are equipped with the same size of cache space. We compare LAM and these methods from the following three aspects, including topology size, cache capacity size and the Zipf distribution parameter α.

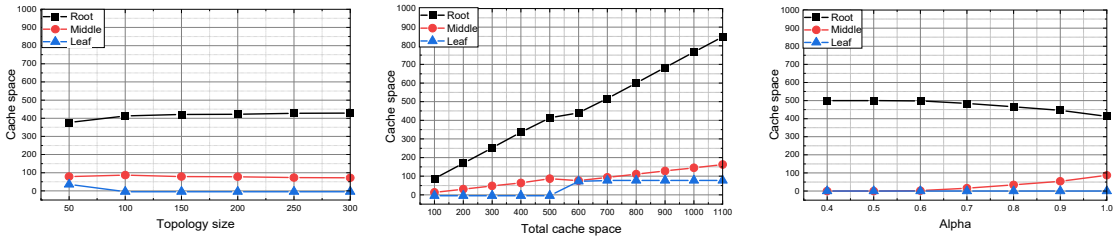
In the content placement scenario, we choose LCD [21] and use it in our caching scheme and the new strategy is denoted as ENC-LCD. In other words, we add the number of content copies control to the original LCD. We compare ENC-LCD with other content placement strategies including Optimal Content Placement (OCP [8]), LCE, original LCD, MCD [22] and CL4M [24]. These six caching strategies are compared in the same three aspects as in the cache allocation scenario. Except for OCP, the LRU cache replacement strategy is equipped with other five caching strategies.

6.2. Comparison in the Cache Allocation Scenario

6.2.1. Cache Space of Different Layers

We choose topologies of three layers for the cache allocation scenario, to observe how the cache space is allocated at different levels. It is obvious that leaf nodes are closer to users. When a request hit happens at a leaf node, network hop count can be minimized for the user. On the other hand, nodes from upper layers can serve more users, so placing the cache in upper layers will promote the cache hit ratio, as well as reducing latency for more users. So it is necessary to allocate cache space to different layers properly.

From Figure 6a, it can be observed that when topology size increases, more cache capacity is allocated to the root node, leading to the cache capacity reduction of middle and leaf nodes. This happens because when the total request number is fixed, requests become more dispersive. Each leaf node and each middle node will serve fewer requests, so placing cache space at the root node brings more benefit. It can be also found the cache capacity of leaf nodes decreases fast than middle nodes.



(**a**) (**b**) (**c**)

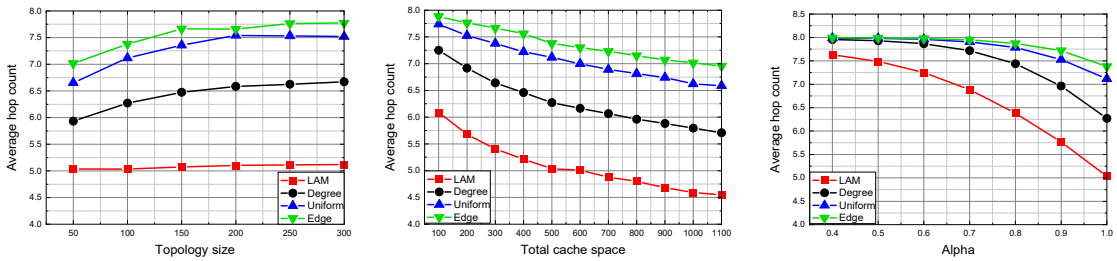
Figure 6. Cache space allocated in different layers. (**a**) The allocated cache space under different topology sizes. (**b**) The allocated cache space under different total cache budget. (**c**) The allocated cache space under different Zipf parameter Alpha.

Figure 6b shows that newly added cache space is allocated to the root node and middle nodes according to a certain proportion. The main reason for placing cache at the root node first is that concentrating cache at the root node can guarantee content diversity. Due to the lack of sharing mechanisms in leaf nodes, caching the same content will consume more space. When the cache space continues to grow, there will be an interval where most newly added space is allocated to leaf nodes (the horizontal axis from 500 to 600 in Figure 6b). This happens because popular content has been cached and caching new content will bring little benefit. Instead, our method decides to allocate cache space to leaf nodes to cache popular content. Later, the newly added space is deployed to the root node and middle nodes according to the same proportion as before.

Figure 6c shows how the Zipf distribution parameter affects the cache space distribution of different layers. As α increases, cache space from the root node continues to decline, while cache space from middle nodes is the opposite. The cache space of middle nodes increases first and then remains stable, while the cache space of leaf nodes keeps steady. This occurs because larger α makes requests from users become more focused on the most popular content. Large cache space in the root node generates little benefit and it will be moved from root nodes to middle nodes to cache popular content, as shown in Figure 6c.

6.2.2. Average Hop Count

We can observe from Figure 7a that, with larger topology sizes, the number of average hop count increases in all methods. As the topology size increases, most nodes get less cache capacity and the distribution of users has become more fragmented. These two reasons make the response of a request occur near the root node or even at the origin server, leading to the increase in average hop count. LAM gets the slowest growth of all methods, meaning that the advantages of LAM become more obvious when topology size increases.



(**a**) (**b**) (**c**)

Figure 7. Performance of average hop count in cache allocation. (**a**) The average hop count under different topology sizes. (**b**) The average hop count under different total cache space. (**c**) The average hop count under different Zipf parameter Alpha.

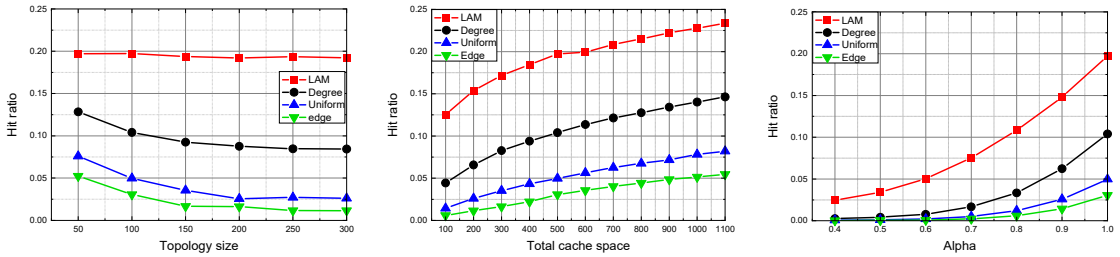
As cache space grows in Figure 7b, the average hop count for all methods comes down slowly. Network performance is improved since more content has been cached. LAM keeps its advantage at different cache space sizes and it performs at least 16.03% lower in average hop count comparing with "Degree".

As α increases in Figure 7c, all methods get lower average hop count. With bigger α, requests become focused on content with high popularity. In this situation, more requests are served at middle nodes and edge nodes and LAM keeps the advantage over other methods.

LAM performs the best at all tested values in different aspects, which is at least 9.04% lower in average hop count comparing with other methods. Based on the analysis above, more cache space should be allocated to the root node and middle nodes when the topology size grows but only LAM and "Degree" follow this pattern. Besides, cache space for middle nodes and edge nodes is positively correlated with Zipf parameter α, only LAM takes advantage of this information. "Degree" only utilizes topology information, so it performs worse than LAM but better than "Uniform". "Edge" performs the worst, which means LCE or caching all content by default is not suitable for "Edge" and the request routing mechanism needs to be adjusted, to enhance the cache sharing rate at leaf nodes. Our method performs on average 18.42% lower in hop count compared with "Degree".

6.2.3. Cache Hit Ratio

As topology size increases, nodes become more dispersed and LAM chooses to put more cache on the root node, as shown in Figure 6a. This makes the cache hit ratio of LAM change not vary significantly in Figure 8a. On the contrary, the cache hit ratio of other methods gradually declines. This shows that LAM performs well at different topology sizes.



(**a**) (**b**) (**c**)

Figure 8. Performance of cache hit ratio in cache allocation. (**a**) The cache hit ratio under different topology sizes. (**b**) The cache hit ratio under different total cache space. (**c**) The cache hit ratio under different Zipf parameter Alpha.

In Figure 8b, as total cache space grows, the network caches more content, so the cache hit ratio increases. And LAM keeps its advantage over all tested values. Compared with "Degree",LAM gets nearly twice the cache hit ratio.

In Figure 8c, as α increases, popular content is more likely to be requested and cached, so all methods get a higher cache hit ratio. LAM adjusts the allocation of cache space according to different α values, so it performs better than other methods. And the advantage of LAM is growing with a bigger α.

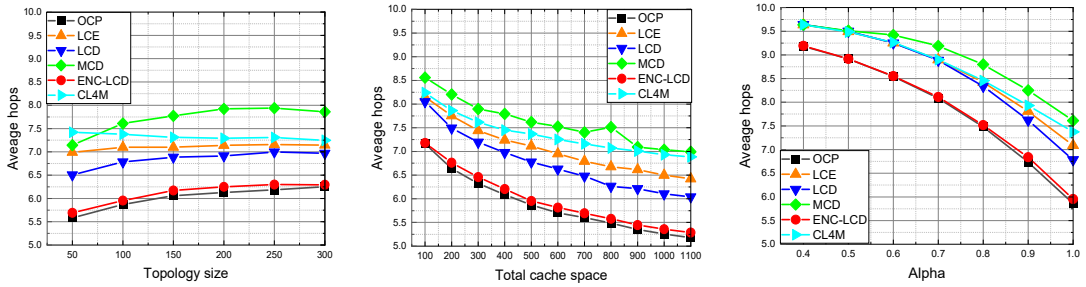
LAM also has an absolute advantage over the cache hit ratio. When α is fixed, LAM tends to allocate more space to the root node when topology size or total cache space grows. This decision guarantees the cache hit ratio to be as high as possible. When α grows, most cache hits happen at middle and leaf nodes, so the gap between different approaches is narrowing. "Degree" lacks content information and has a lower cache hit ratio. "Edge" lacks cache sharing mechanism among leaf nodes and its cache hit ratio is the lowest. In the cache allocation scenario, our method performs 57.88% higher in cache hit ratio compared with "Degree".

6.3. Comparison on Content Placement

We compare ENC-LCD with OCP, LCE, LCD, MCD and CL4M, while the cache allocation scheme is the same. ENC-LCD adds the number control of content copies to LCD, while OCP uses both ENC and the network position information. To apply OCP strategy in ICN, one way is to pre-deploy content to corresponding space on nodes. Another way is to notify each node of what content to cache and nodes record messages in their own solution lists. In the experiment, we use the second way to apply the OCP strategy.

6.3.1. Average Hop Count

From Figure 9a, as topology size grows, average hop count from most strategies increases except CL4M. Figure 9b,c show that when total cache space or α increases, the average hop count from all strategies decreases and gaps among them are stable. Compared with LCD, the average hop count of ENC-LCD is 9.96% lower. In different aspects, OCP keeps the lowest average hop count, for the reason that the content placement solution takes into account the impact of position information and content information. ENC-LCD performs a little behind OCP but is clearly ahead of other strategies. MCD performs the worst and LCD has a slight advantage over LCE and CL4M. OCP searches for the optimal placement from the perspective of the network, so lower average hop count is reached. MCD will delete the original copy when placing a new copy downstream, which reduces the content share rate and increases the average hop count.



(**a**) (**b**) (**c**)

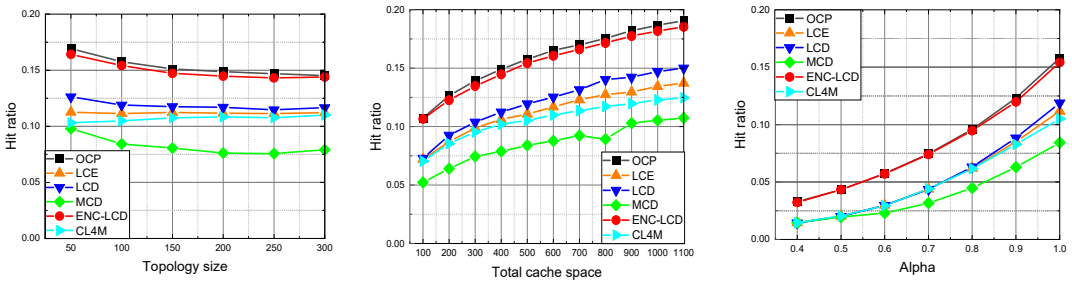
Figure 9. Performance of average hop count in content placement. (**a**) The average hop count under different topology sizes. (**b**) The average hop count under different total cache space. (**c**) The average hop count under different Zipf parameter Alpha.

6.3.2. Cache Hit Ratio

Figure 10 shows that in different aspects, OCP keeps the highest cache hit ratio and ENC-LCD gets the second place. These two strategies use the message of ENC to reduce content redundancy and improve content diversity, so they bring in a higher cache hit ratio. MCD performs the worst, while LCE, LCD and CL4M reflect the same trend of change. When topology size grows in Figure 10a, other strategies have a stable cache hit ratio, while that of MCD is increasing. From Figure 10b, more cache space leads to a higher cache hit ratio of all strategies and ENC-LCD has a cache hit rate of about 0.1 higher than the LCD. In Figure 10c, when α increases, all strategies have a higher cache hit ratio and the gap between OCP and the second-best strategy is increasing. ENC-LCD performs on average 31.54% higher in the cache hit ratio compared with LCD.

From the above analysis, we can find that caching strategies will significantly affect network performance. As the default cache strategy in ICN, LCE can be further optimized. LCD can reduce network hops meanwhile keeping the cache hit ratio as LCE. MCD and CL4M, on the contrary, are not suitable for the ICN hierarchy network. OCP places content according to the near-optimal solution and it shows good performance in different aspects. ENC-LCD performs close to OCP and it will not bring in extra network traffic. Besides, ENC caching scheme has stronger scalability. When the content

set is changing dynamically, all nodes using OCP have to update their solution lists frequently. On the contrary, the ENC-based caching scheme leaves caching decisions to the on-path nodes. These nodes take advantage of ENC of content and other information and they can make diverse decisions based on different scenarios.



(**a**) (**b**) (**c**)

Figure 10. Performance of cache hit ratio in content placement. (**a**) The cache hit ratio under different topology sizes. (**b**) The cache hit ratio under different total cache space. (**c**) The cache hit ratio under different Zipf parameter Alpha.

7. Consideration

7.1. Arbitrary Topology

With the proposed LAM and ENC-based caching scheme, network average hop count gets a significant decrease in hierarchy topologies, while performance in arbitrary topologies needs further verification. When topology becomes complicated, it is difficult to compute the benefit of content. Some graph theory methods are required to extract more information from the network.

7.2. Multi-Objective Optimization

In this paper, hop count from requests is used to evaluate the network performance in ICN caching scenario. Apart from hop count, several features are also important and can reflect the characteristics of the ICN network, such as power consumption and content diversity, which also reflect the network performance from different aspects. In Reference [30], caching efficiency was formed as a multi-objective optimization problem with constraints, to evaluate the impact of different parameters, such as file size, copy size and the probability of replacement. For an Internet service provider (ISP), to minimize the cost and to maximize the network performance like the cache hit ratio, different kinds of multi-objective optimization problems would be founded [31].

7.3. Dynamically Changing Content

We assume the content popularity is certain for a period of time in this paper. However, content popularity usually changes over time. Some measuring mechanisms are required to analyze content popularity. And cache allocation and content placement solution should be adjusted based on changes in content popularity.

8. Conclusions

In this paper, we address the cache allocation problem and the content placement problem in the ICN hierarchical network under a given cache space budget. We propose a Light Weight Method (LAM) to allocate cache space properly, as well as generate Expected Number of Copies (ENC) of popular content. An on-path caching scheme based on ENC is raised for content placement. Experiment results show that in the cache allocation scenario, LAM has the lowest hop count and highest cache hit ratio compared with "Degree Centrality" allocation method and "Edge Caching" method. In the

content placement scenario, the proposed ENC-LCD under the on-path caching scheme brings further improvement compared with LCE, LCD, MCD and CL4M. Our future direction will focus on this on-path caching scheme under arbitrary network topology and with a changing content set, as well as multi-objective optimization of the ICN network.

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 Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100).

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 no conflict of interest.

References

1. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 1024–1049. [CrossRef]
2. Ravindran, R.; Chakraborti, A.; Amin, S.O.; Azgin, A.; Wang, G. 5G-ICN: Delivering ICN services over 5G using network slicing. *IEEE Commun. Mag.* **2017**, *55*, 101–107. [CrossRef]
3. Hajimirsadeghi, M.; Mandayam, N.B.; Reznik, A. Joint caching and pricing strategies for information centric networks. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015; pp. 1–6.
4. Dabirmoghaddam, A.; Barijough, M.M.; Garcia-Luna-Aceves, J.J. Understanding optimal caching and opportunistic caching at the edge of information-centric networks. In Proceedings of the 1st ACM Conference on Information-Centric Networking, Paris, France, 24–26 September 2014; pp. 47–56.
5. 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, Helsinki, Finland, 17 August 2012; pp. 55–60.
6. Xu, Y.; Li, Y.; Lin, T.; Wang, Z.; Niu, W.; Tang, H.; Ci, S. A novel cache size optimization scheme based on manifold learning in content centric networking. *J. Netw. Comput. Appl.* **2014**, *37*, 273–281. [CrossRef]
7. Rossi, D.; Rossini, G. On sizing CCN content stores by exploiting topological information. In Proceedings of the 2012 IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 March 2012; pp. 280–285.
8. Azimdoost, B.; Farhadi, G.; Abani, N.; Ito, A. Optimal in-network cache allocation and content placement. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; pp. 263–268.
9. Wang, Y.; Li, Z.; Tyson, G.; Uhlig, S.; Xie, G. Optimal Cache Allocation for Content-Centric Networking. In Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 7–10 October 2013; pp. 1–10.
10. Breslau, L.; Cao, P.; Fan, L.; Phillips, G.; Shenker, S. Web caching and Zipf-like distributions: Evidence and implications. In Proceedings of the IEEE INFOCOM 1999, New York, NY, USA, 21–25 March 1999; pp. 126–134.
11. Zhang, G.; Li, Y.; Lin, T. Caching in information centric networking: A survey. *Comput. Netw.* **2013**, *57*, 3128–3141. [CrossRef]
12. Chu, W.; Dehghan, M.; Towsley, D.; Zhang, Z.L. On allocating cache resources to content providers. In Proceedings of the 3rd ACM Conference on Information-Centric Networking, New York, NY, USA, 26–28 September 2016; pp. 154–159.
13. Cui, Y.; He, W.; Ni, C.; Guo, C.; Liu, Z. Energy-efficient resource allocation for cache-assisted mobile edge computing. In Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Singapore, 9–12 October 2017; pp. 640–648.

14. Ndikumana, A.; Ullah, S.; LeAnh, T.; Tran, N.H.; Hong, C.S. Collaborative cache allocation and computation offloading in mobile edge computing. In Proceedings of the 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), Seoul, Korea, 27–29 September 2017; pp. 366–369.
15. Huo, R.; Xie, R.; Zhang, H.; Huang, T.; Liu, Y. What to cache: Differentiated caching resource allocation and management in information-centric networking. *China Commun.* **2016**, *13*, 261–276. [CrossRef]
16. Ding, Q.; Pang, H.; Sun, L. SAM: Cache space allocation in collaborative edge-caching network. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
17. Banerjee, B.; Kulkarni, A.; Seetharam, A. Greedy Caching: An Optimized Content Placement Strategy for Information-centric Networks. *Comput. Netw.* **2018**, *140*, 78–91. [CrossRef]
18. Fayazbakhsh, S.K.; Lin, Y.; Tootoonchian, A.; Ghodsi, A.; Koponen, T.; Maggs, B.; Ng, K.C.; Sekar, V.; Shenker, S. Less Pain, Most of the Gain: Incrementally Deployable ICN. *Comput. Commun. Rev.* **2013**, *43*, 147–158. [CrossRef]
19. Mehran, N.; Movahhedinia, N. Non-uniform EWMA-PCA based cache size allocation scheme in Named Data Networks. *Chin. Sci.* **2018**, *61*, 119–131. [CrossRef]
20. Duan, J.; Xing, Y.; Zhao, G.; Zeng, S.; Liu, Y. Fine-grained cache deployment scheme for arbitrary topology in ICN. *Neural Comput. Appl.* **2019**, *31*, 3357–3368. [CrossRef]
21. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634. [CrossRef]
22. Laoutaris, N.; Syntila, S.; Stavrakakis, I. Meta algorithms for hierarchical web caches. In Proceedings of the IEEE International Conference on Performance, Computing, and Communications, Phoenix, AZ, USA, 15–17 April 2004; pp. 445–452.
23. 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.
24. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache "less for more" in information-centric networks. In Proceedings of the International Conference on Research in Networking, Berlin/Heidelberg, Germany, 21–25 May 2012; pp. 27–40.
25. Khan, J.A.; Westphal, C.; Ghamri-Doudane, Y. A Popularity-aware Centrality Metric for Content Placement in Information Centric Networks. In Proceedings of the Computing, Networking and Communications (ICNC), Maui, HI, USA, 5–8 March 2018; pp. 554–560.
26. Applegate, D.; Archer, A.; Gopalakrishnan, V.; Lee, S.; Ramakrishnan, K.K. Optimal content placement for a large-scale VoD system. *IEEE/ACM Trans. Netw.* **2015**, *24*, 2114–2127. [CrossRef]
27. Al-Fares, M.; Loukissas, A.; Vahdat, A. A scalable, commodity data center network architecture. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 63–74. [CrossRef]
28. Shi, T.; Xu, Y.; Zhao, Z. Coordinated Content Caching and Request Routing Algorithm in Information Centric Networking. *J. Netw. New Media* **2014**, *3*, 26–31.
29. Chesire, M.; Wolman, A.; Voelker, G.M.; Levy, H.M. Measurement and analysis of a streaming media workload. In Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, CA, USA, 26–28 March 2001; pp. 1–12.
30. Liu, W.X.; Yu, S.Z.; Gao, Y.; Wu, W.T. Caching efficiency of information-centric networking. *IET Netw.* **2013**, *2*, 53–62. [CrossRef]
31. Araldo, A.; Mangili, M.; Martignon, F.; Rossi, D. Cost-aware caching: Optimizing cache provisioning and object placement in ICN. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014; pp. 1108–1113.