future internet

MDPI

Article

A Hierarchical Cache Size Allocation Scheme Based on Content Dissemination in Information-Centric Networks

Hongyu Liu [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; liuhy@dsp.ac.cn
[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

Abstract: With the rapid growth of mass content retrieval on the Internet, Information-Centric Network (ICN) has become one of the hotspots in the field of future network architectures. The in-network cache is an important feature of ICN. For better network performance in ICN, the cache size on each node should be allocated in proportion to its importance. However, in some current studies, the importance of cache nodes is usually determined by their location in the network topology, ignoring their roles in the actual content transmission process. In this paper, we focus on the allocation of cache size for each node within a given total cache space budget. We explore the impact of heterogeneous cache allocation on content dissemination under the same ICN infrastructure and we quantify the importance of nodes from content dissemination and network topology. To this purpose, we implement a hierarchy partitioning method based on content dissemination, then we formulate a set of weight calculation methods for these hierarchies and to provide a per-node cache space allocation to allocate the total cache space budget to each node in the network. The performance of the scheme is evaluated on the Garr topology, and the average hit ratio, latency, and load are compared to show that the proposed scheme has better performance in these aspects than other schemes.

Keywords: Information-Centric Networking (ICN); in-network caching; cache size allocation; content dissemination

Citation: Liu, H.; Han, R. A Hierarchical Cache Size Allocation Scheme Based on Content Dissemination in Information-Centric Networks. *Future Internet* **2021**, *13*, 131. https://doi.org/10.3390/fi13050131

Academic Editor: Paolo Bellavista

Received: 8 April 2021
Accepted: 13 May 2021
Published: 15 May 2021

1. Introduction

The traditional network architecture has strongly supported the vigorous development of the Internet over the years. However, it has become gradually insufficient [1] as the current usage of the Internet is increasingly focused on content retrieval and dissemination, while its transmission architecture, proposed in the 1960s, is still based on the end-to-end communication paradigm. The mismatch between the location-based routing and the distributed retrieval schemes is the basic reason for several significant insufficiencies of the Internet [2]. Such as security, mobility management, and scalability. Based on the above background, the future networks proposed to overcome the shortcomings of the traditional Internet has become a research hotspot, and ICN is considered one of the most influential solutions in many scenarios [3]. The primary difference between ICN and the existing Internet is that the former focuses on the content itself rather than its location. The shift from host-to-host to content-centric in ICN provides a natural solution to most of the issues faced by the IP infrastructure today, which are accrued by current Internet trends. The current representative ICN architectures are Data-Oriented Network Architecture (DONA) [4], Publish/Subscribe Internet Routing Paradigm (PSIRP) [5], Network of Information (NetInf) [6] and MobilityFirst [7,8], Content-Centric Networking (CCN) [9] in Named Data Networking (NDN) project [10]. In addition, some are designed to smoothly transfer from the current network to ICN [11]. The first-class citizen in ICN is the content,

not the end host. By naming content at the network layer, ICN supports name-based routing, which decouples routing from the location. Furthermore, all of the candidate ICN architectures share a fundamental principle of universal in-network caching. In-network caching, one of the most essential features of ICN, is used to store data packets for future retrieval [12], and its benefits can be roughly divided into three categories [13]: From the perspective of Internet Service Provider, the in-network caching can resolve the requests for stored content, so those content will not be repeatedly transmitted in the core network. By offloading the core network traffic to each autonomous system, the traffic load on the core network is greatly reduced. From the perspective of the content provider, the traffic load on the content server can also be reduced, because the in-network caching prevents the requests to the content server. From the perspective of the user, in-network caching is closer to users compared with content providers, which reduces the latency for users to retrieve content objects. The benefits that in-network caching brings are particularly similar to those of the content distribution network (CDN) [14]. And the success of commercial CDN proves the effectiveness of in-network caching.

Improving caching performance is a hot issue in the current related research fields of ICN. To realize in-network caching, network nodes (nodes and routers are not distinguished in this paper) are equipped with content storage composing the caching network. The cache is only equipped on specific nodes in CDN, and its access rate is not high. Thus, low-cost hard disks can be used as storage media, which makes the content storage space almost infinite. However, ICN requires line-speed execution for caching and therefore has higher requirements for storage media. On the one hand, these high-speed storage media are expensive; on the other hand, the maximum capacity of these high-speed storage media is limited. Consequently, the total amount of cache in ICN is limited. A reasonable cache size allocation scheme can make the best of the limited cache space to result in better performance.

The goal of our work is to make the most of the given total cache budget by allocating it to each node in the network effectively. The cache size allocation scheme proposed in this paper adopts a proportional distribution method, that is, the size of the cache space of each node is proportional to its importance. The existing proportional division methods are mainly based on the network topology but ignore the content transmission process. The consequent mismatch in terms of communication semantics between "where" (location of the node) and "how" (the content dissemination) is at the origin of a number of significant deficiencies of these schemes. Hence, we analyze the actual transmission process of content in given network topology, then calculate the importance of each node according to its role in the actual transmission, and finally, allocate the given total cache space to each network node. The main contributions of this paper are as follows:

- We implement a hierarchy partitioning method based on content dissemination to simplify the network topology with the arbitrary graph structure, while the content transmission process is clarified.
- We quantify the importance of nodes from the content transmission process and network topology by formulating a set of calculation methods for weights at each level, which resolves the mismatch between content dissemination and topology location.
- We use real-world network topology to evaluate the performance of the cache size allocation scheme proposed. The results show that our scheme performs better in different aspects.

The rest of this article is organized as follows. In Section 2, we introduce the background and related work of ICN caching. Section 3 is the problem statement. In Section 4, we introduce the basis and method of network hierarchy division. In Section 5, we introduce the calculation method of weight and the specific allocation method of cache. Section 6 gives the results of quantitative experiments. We discuss this in Section 7. Finally, we summarize the paper in Section 8.

2. Background and Related Work

2.1. New Features and Challenges

There are three main differences between caching in ICN and existing caching systems:

Transparency, the traditional caching is usually in the form of a dedicated closed system, mainly for a single traffic type. For example, objects are logically isolated by domain in cache systems such as Web and CDN. In ICN routing and storage based on the globally unique identifier of the content completely realizes the independence of caching and application, making it a basic service that is universal, open, and transparent to the upper layer. At the same time, caching transparency also brings problems such as inconsistent caching targets and cross-application competition for caching resources.

Universality, caching is generally only equipped with specific nodes in traditional cache research. The network structure of the cache is a regular tree-like hierarchy, so the mathematical model of collaboration and content placement between caches can be established based on a priori knowledge of traffic requirements and cache structure, then the optimal solution can be obtained. In ICN, the cache can be equipped on any node. As a result, the topology of the cache needs to be described by the mesh structure of an arbitrary graph, and the upstream and downstream relationship between nodes is no longer clear. The difficulty of mathematical modeling and analysis has increased.

Fine granularity, the objects stored and replaced in traditional caching systems are generally based on files or variable segments of files [15–17]. However, such caching systems have a large read and write overhead, which is difficult to meet the requirement of the ICN [18,19]. Therefore, the file in ICN is divided into smaller independently identifiable data blocks (chunks) which are smaller than those in CDN and so on [20], and chunks are the basic caching units [9,18,21]. This makes many analysis reference models of traditional cache invalid for ICN.

2.2. Researches in Caching Field

2.2.1. Present Research Situation

Because of these new features and challenges, there is a wide variety of research on ICN caching. At present, these research works are roughly divided into two categories as shown in Figure 1: one is to optimize the performance of the cache network system; the other is to establish and analyze the theoretical model of the cache network. The former tends to be specific and optimizes the overall performance of the cache system from different perspectives. The latter is to abstract and simplify the network to improve the theoretical support for understanding the behavior of the network system.

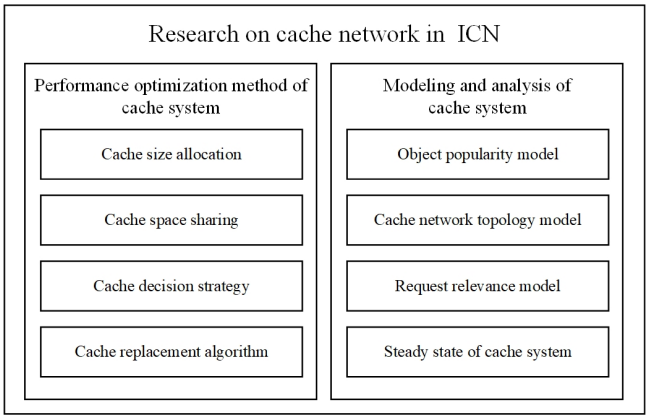


Figure 1. Present Research on Cache in ICN.

While less work has been done on cache size allocation in existing studies [22], this does not mean that it is unimportant. The size of the cache storage space will undoubtedly affect the performance of the entire cache system. When the other settings of the system are the same, the nodes with larger cache size can cache more objects, which will improve the

cache hit ratio for the entire system. At the same time, the cost of query caching increases as the cache space increases. ICN requires nodes to execute at line speed, which limits the maximum cache size that a node can support. To optimize the performance of the cache network from the perspective of cache size, the main problem to be solved is the allocation of cache resources in different nodes. That is, which nodes should be allocated more cache space within the total cache budget to improve the overall performance of the cache system.

2.2.2. Cache Size Allocation Scheme

Cache size allocation determines the efficiency of many cache studies, and researchers have done a lot on it. Homogeneous cache allocation is the default cache size allocation scheme in ICN, distributing the total cache budget equally to all nodes. Many other studies in ICNs, such as cache placement, replacement, and routing forwarding, are designed based on the assumption of homogeneous cache allocation [23,24]. That scheme is simple and easy, but ignores the differences in the importance of nodes in content distribution, resulting in an insignificant effect. To solve the limitations of homogeneous cache allocation schemes, relevant researchers have proposed some improvement strategies. These studies are divided into two main categories.

One is proportional allocation. Such studies typically analyzed the importance of nodes and then allocated the size of the cache proportional to the importance. For example, reference [25] considered such indicators as degree centrality, pressure centrality, betweenness centrality, proximity centrality, and graph centrality to design different cache size allocation schemes. Reference [26] used Manifold Learning to analyze network traffic characteristics and user behavior, classified routers according to their role in content distribution, and assigned cache sizes to nodes according to their categories. However, this scheme relied on complex automatic learning algorithms, and it was difficult to distinguish the importance of cache nodes in a new CCN environment. Reference [27] defined a new measure, request impact degree (RID), that reflected the importance of routers on the delivery path of content requests. Then they proposed a scheme to allocate the size of cache space in proportion to RID. Reference [28] analyzed how to allocate storage space for routers in a given storage budget network. By establishing a mathematical model to solve the problem, they found that network topology, network size, content popularity, and other factors would have some impact on cache allocation, so there was no one strategy for all cases. Reference [29] was the first study on cache size allocation, but it was designed for traditional networks with the purpose of selecting the appropriate cache location to reduce network traffic. Nevertheless, this also provided a solid foundation for future researches on cache space placement in ICN. Some studies considered that network structure had distinct community characteristics and the cache on nodes mainly satisfied local user's demand for content. Using the characteristic spectrum of network adjacency matrix to reflect the number of communities in the network [30], the researchers proposed that the importance of the node community should be used to represent the local importance of the node, and then determined the size of the node cache space in ICN. There are some other studies that considered both the network topology information and the dynamic changes of user demand. A new measure called node weight was defined, and a cache size allocation scheme based on node weight was designed.

On the other hand, researchers designed optimization problems for some network performance, and continuously simplified and solved them. Reference [31] established an optimization model to study the migration of ICN under budget constraints from an economic perspective. There were also studies established three optimization models from the perspectives of users, ISPs, and content providers. They solved the three optimization problems and intersected their solutions to determine the location selection of cache nodes. Reference [32] established an optimization problem to determine the optimal storage allocation and content placement in a hierarchical topology with total storage budget constraints. The scheme also considered capacity constraints for downlinks and uplinks

used to allocate storage to minimize data transmission costs. This solution applied to both wireless and wired networks. Reference [33] found the key factors that affect the performance of CCN caches and presented an optimization scheme to calculate how many caches those CCN nodes should be allocated to the configuration cache.

There were also some studies with unique perspectives. Reference [34] proposed a lightweight allocation method utilizing information of both topology and content popularity to allocate cache space and get the expected number of copies of popular content. When the cache is deployed, dramatic changes are not likely to occur in the short term, while the network is constantly changing. To solve of this contradiction, some studies proved the rationality of cache debit and proposed a dynamic debit mechanism for cache space. They lent part of the space on free nodes to busy nodes to improve the overall utilization of cache space on nodes. To simplify the complex factors that affect the cache performance of the network, reference [35] used the t-SNE (t-distributed Stochastic Neighbor Embedding) algorithm to reduce the dimensionality of the cached feature dataset which presented high-dimensional features. The result was then used to cluster nodes and calculate the weights of various nodes. Finally, nodes were allocated cache capacity space according to their weights.

3. Problem Statement

In this section, we introduce a network model of an arbitrary graph, analyze the cache size allocation problem, and point out the limitations of existing schemes, which is the main problem to be solved in this paper.

3.1. System Model

We choose a topology with arbitrary graph structure as shown in Figure 2 to introduce our model. When modeling a traditional network, a tree-shaped structure is usually used to represent the network, which has clear hierarchical and upstream and downstream relationships. Generally, there is only one root node of this structure to facilitate network analysis. In the actual ICN scenario, root nodes are not unique and connected to each other, and the ubiquity of the cache obscures the upstream and downstream relationships between nodes, so an arbitrary graph topology should be used to represent the ICN structure more appropriately.

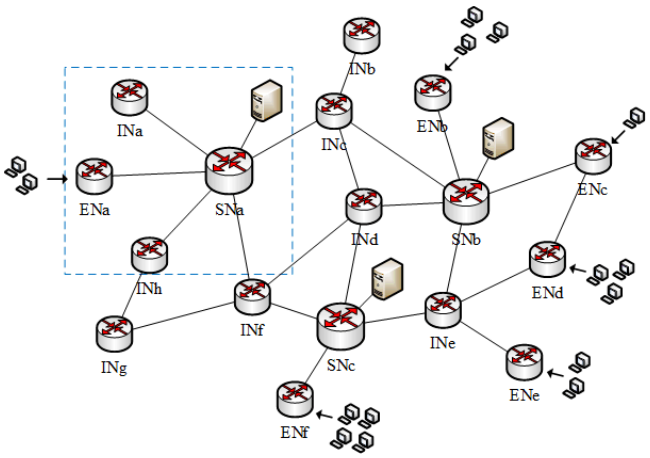


Figure 2. The Arbitrary Graph Structure Network Model.

The network topology shown in Figure 2 consists of three source nodes represented as SNa, SNb and SNc connected to the content servers, several edge nodes represented as ENa, ENb, ENc, etc. connected to users, and intermediate nodes represented as INa, INb, INc, etc. connected only to other nodes but not directly to users or servers. It is important to note that an arbitrary graph topological network of any size still applies to this model. In this model, factors such as the number and location of content servers or

users and network structure only affect the allocation results as input parameters, but not the allocation scheme itself. Therefore, this model can easily be extended to a larger arbitrary graph structure network. Table 1 represents the main notations used in this paper.

Table 1. Experiment parameters.

Parameters	Value
CS	Content storage on nodes
C_i	Content i in the network or request for content i
CDW_i	Weight of content domain i
DPW_i	Weight of data path i
NW_i	Weight of node i
NG_n	Global weight of node n
CD	Collection of content domains in a network
DP	Collection of data paths in a content domain
N	Collection of nodes in a data path

In this paper, to facilitate analysis, we make the following assumptions:

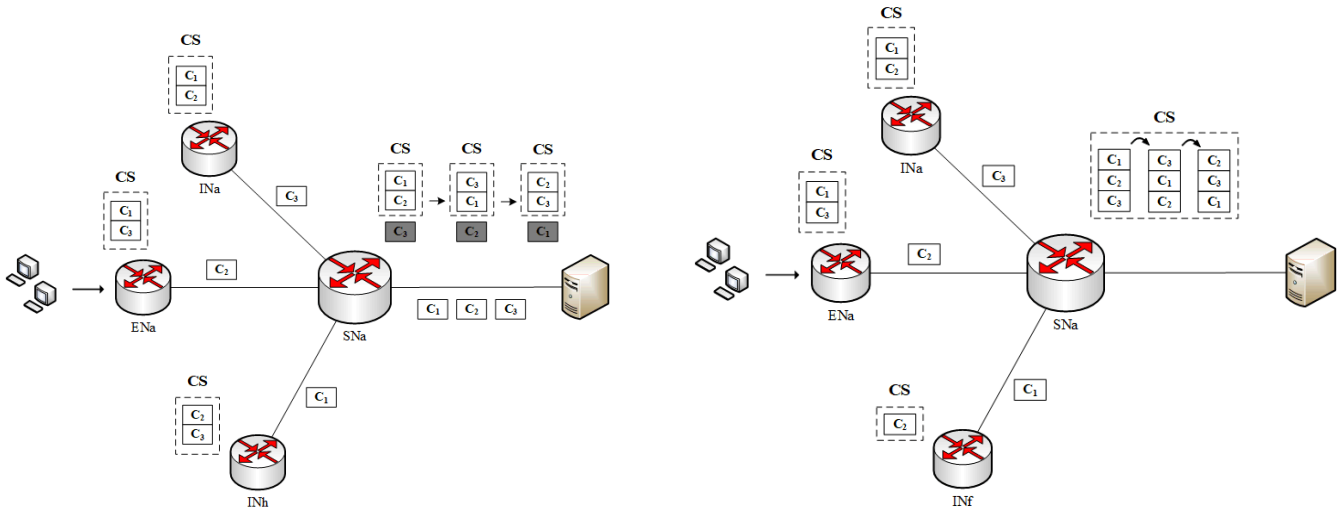
The three content servers store all the content for retrieval.

Each node in the network is able to be equipped with cache storage to cache content. When a node stores the requested content, it responds to the request and no longer forwards it to the server; otherwise, the node forwards the request to the next node along the shortest path.

This network model uses an on-path caching strategy, requests are forwarded along the shortest path, and content is forwarded along the same path as requests.

3.2. Problem Analysis

The homogeneous cache allocation scheme ignores the differences in the importance of nodes in network topology and content distribution. While this default scheme is simple and easy, it does not make full use of the limited total cache budget, which seriously affects the performance of the cache system. Figure 3 compares homogeneous and heterogeneous cache allocations using a portion of the network model.



(**a**) Homogeneous cache allocation scheme (**b**) Heterogeneous cache allocation scheme

Figure 3. Comparison of Two Caching Schemes.

Figure 3a uses a homogeneous allocation scheme, with four nodes of the same size of storage space, all set up as two storage units. The request from INa for content C_3 was

forwarded to SNa for further processing and forwarding because it missed on INa. SNa also does not store C_3, so the request is forwarded to the server. After the server responds to the request, C_3 in the storage units of INa and SNa replaces the original C_2, which causes subsequent ENa requests for C_2 to also be sent to the content server for processing. This makes C_1 requests forwarded by INf have to be sent to the server in response. The cache loss on SNa is caused by frequent replacements of content due to the insufficient number of storage units. In the heterogeneous allocation scenario, as shown in Figure 3b, a storage unit of INf is allocated to SNa, where C_1, C_2, and C_3 can all be responded to in SNa, and the cache hit ratio is significantly improved.

Therefore, heterogeneous allocation is necessary to improve the performance of the cache system. Nodes that play a high role in content distribution should be allocated more cache sizes. In the example above, INa, ENa, and INf forward their own missed requests to SNa for subsequent processing and forwarding. SNa plays a significantly higher role in content distribution than other nodes. Allocating more cache space to SNa can significantly improve the cache hit ratio. The actual network structure is very complex, and it is unrealistic to analyze each node like the example above. Then, the key to the problem is to assign the nodes in a heterogeneous way.

Current schemes are often based on the topological structure of the network, such as degree-centricity which represents the local importance of nodes, and betweenness centrality which represents the global importance of nodes. Network topology is bound to have an important impact on content dissemination, but in many scenarios, the importance of nodes in the topology is not exactly the same as that in content dissemination. The betweenness centrality of nodes in the model is shown in Figure 4.

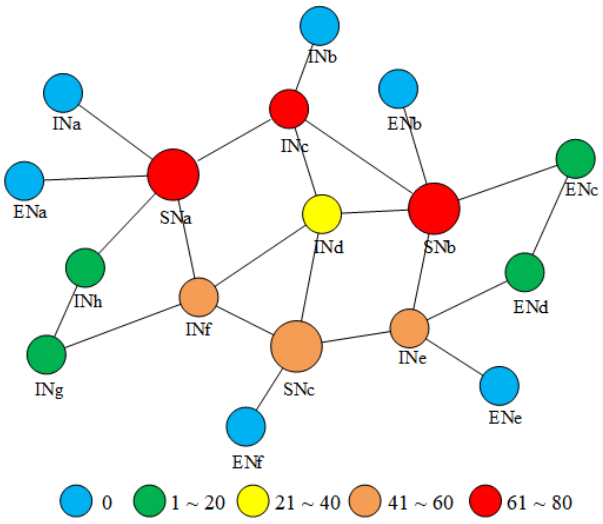


Figure 4. Betweenness Centrality of Network Nodes.

The betweenness centrality of each node is the number of times the shortest path between any two points in the graph passes through the node. In Figure 4, the values of the betweenness centrality from blue, green, yellow, orange to red gradually increase. Nodes with higher betweenness centrality values are mainly located in the center of the network. INc, SNa, and SNb have a high betweenness centrality, but in the actual content distribution process, these nodes are not the nodes with the most paths, which limits the efficiency of the cache size allocation method proportional to the betweenness centrality under cache size conditions. Other schemes based on a network topology that ignore the content distribution process also have these shortcomings.

While schemes such as dynamic debit caching can further improve the performance of the cache system after the cache is deployed, the size of the content in the network usually exceeds the size of the cache space. When the network reaches a steady state, each node should be full of content and implement a cache replacement strategy, making the dynamic

debit scheme useless. Therefore, reasonable planning of the total cache budget before cache deployment can make more efficient use of the limited cache resources. The hierarchical cache size allocation scheme proposed in this paper is to heterogeneously allocate the cache size in combination with the content distribution process. This cache size allocation scheme allows planning the allocation of the cache before network deployment to determine how much cache space each node should be allocated. The data required for parameter calculation is collected and stored by each node, and the processing of the data is offline on additional servers. In the range of adjustable disk space of ICN nodes, it can be dynamically adjusted according to the algorithm. When the allocation scheme changes greatly, the cache space of the nodes needs to be expanded according to the algorithm.

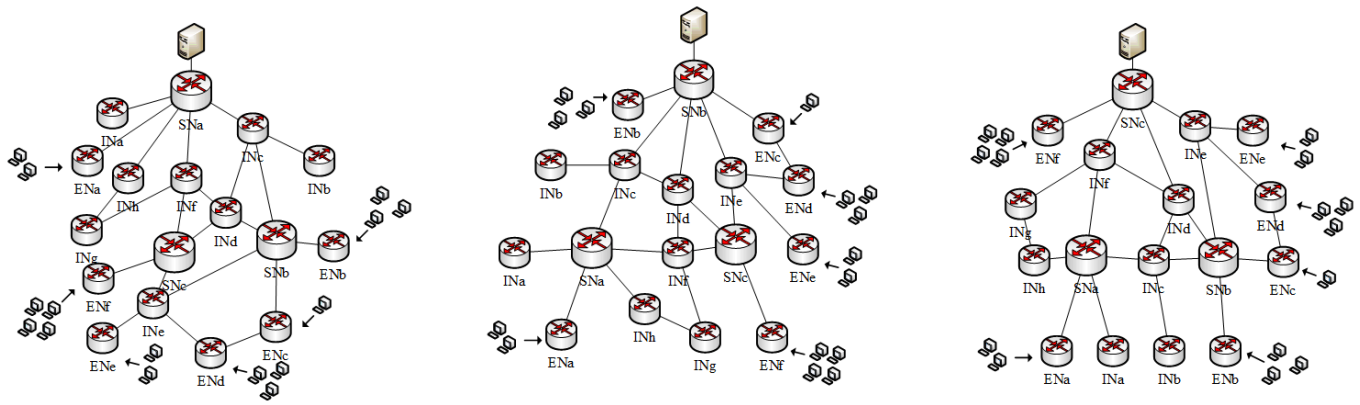
4. Level Division

The arbitrary graph structure of the ICN makes network analysis more difficult. Inspired by the traditional hierarchical tree structure analysis, we divide the arbitrary graph structure of ICN into three levels: content domain, data path, and node from the perspective of content dissemination to simplify the network. This is a logical division where different domains and paths are not physically isolated, in detail, the same node can belong to several different domains and paths at the same time. This hierarchical network partitioning not only simplifies complex networks but also dissects the content dissemination processes and resolves the inaccurate importance assessment of nodes due to mismatches between their location in the topology and their role in content transmission.

4.1. Content Domain

The mass content acquisition is the most important feature of ICN. With the emergence of more and more large content providers, content and applications on the network tend to be diversified. Transparency of in-network caches in ICN decouples the application from the cache, allowing different types of content from different content servers to be stored in the same node's cache space. This makes the analysis of the cache system very difficult. So first we divide the network into different content domains.

Cached content is ubiquitous throughout the network. But the source of content, the content server, cannot be placed anywhere in the network. Not all nodes have the access capabilities required by the content server. Therefore, in a deterministic network, the location where the content server can be placed is deterministic. Based on this, we separately divide the content domain at the root of each source node connected to the content server. In the model established in the previous chapter, SNa, SNb, and SNc are connected to the content server respectively. We divide the network into three content domains using these three nodes as roots. The content domains are shown in Figure 5.



(**a**) Content Domains A (**b**) Content Domains B (**c**) Content Domains C

Figure 5. The Content Domains.

This partitioning allows only one node in each domain to connect to the content server. This one-node structure is similar to the traditional tree structure. Each content domain has intersections with other content domains that contain the same nodes. Since there is only one root node in a content domain, the domains are logically isolated from each other. This division shifts the objects we analyze from a network with complex structure and diverse content to several content domains with simple structure and single content, enabling us to see more clearly the impact of content distribution processes and caching on them.

4.2. Data Path

Passive caching is used in ICNs where the node stores the requested content that passes through it, rather than pre-storing some content on the node. This allows the caching system to cache current hot spots and eliminate outdated content adaptively to network access. But it also brings some uncertainty to cache research. Also, content name routing is supported in ICN, so forwarding no longer depends on the location of the content. As a result, we do not know at which node the request will be answered or will eventually arrive at the content server to respond. To solve these problems, we set up a special scenario for analysis: assuming that neither node in the network caches anything, or that the content of this request is not cached in any node in the network.

In this case, no node in the network can respond to the received request, and the request will be forwarded to the next node by each node it passes through.

With on-path caching, requests are forwarded by the shortest path from the user to the content server, each node along the way tries to respond to them. Finally, requests are forwarded to the content server.

With an off-path caching strategy, requests access each node in turn according to the HASH calculation results until the request reaches the content server.

Caching makes it uncertain which node responds to requests, but the direction of request forwarding is determined, and requests are forwarded to the content server regardless of the caching strategy. For analysis purposes, this paper considers only scenarios with on-path caching strategies, further dividing the content domain into data paths as shown in Figure 6.

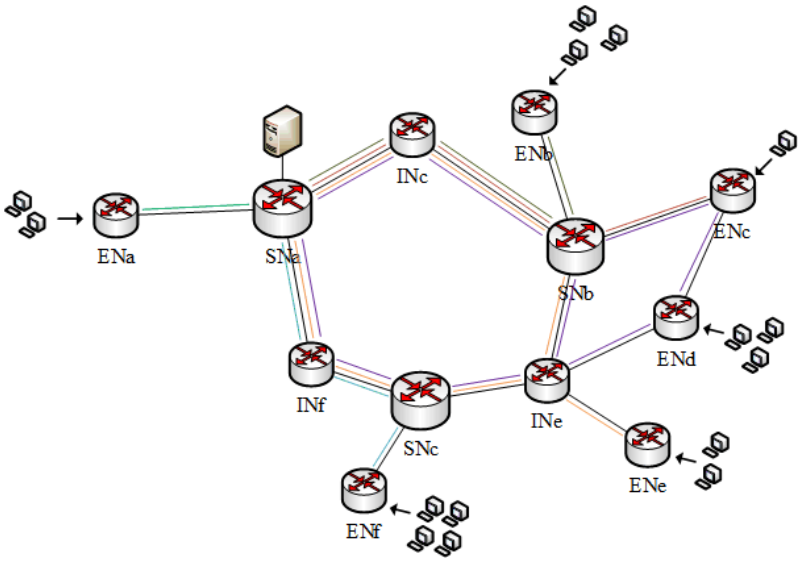


Figure 6. The Data Paths in Content Domain A.

Figure 6 divides content domain A into eight data paths, each of which is the shortest path from the user to the content server. If there are several shortest paths between the access node and the source node, each constitutes a data path. Different data paths may also contain the same node, but since requests on each path come from different users, they

are logically isolated. These data paths clearly show the forwarding paths of requests and content in the network.

4.3. Node

By dividing the network into content domains and further into data paths, the relationship between network nodes has changed from intricate to simple. The upstream and downstream relationships between nodes become clear. Cache size allocation requires that the total cache budget be allocated to a specific node, so the node is the smallest unit of study in this paper.

5. Cache Allocation

The process of content distribution is demonstrated through hierarchical division. We will do further analysis at different levels, calculate the weights of each level, and then allocate the caches proportionally to each level until they are allocated to each node in the network.

5.1. Content Domain Weight

Content domains logically separate content from different content servers. The size of contents is the most important element in a content domain and the biggest difference between different content domains. The size of the content intuitively affects the adequacy of cache space, so we first considered the content size. The weight of a content field is determined by what it contains. The more content a domain contains, the more cache size it should be allocated, so its weight should also be greater. However, different categories of content with the same size, such as VoD and Web, require different cache space to achieve the same cache hit ratio, which is represented by cache size sensitivity. In addition, more caches should be allocated to those hot spots because they are more likely to be requested.

Combining the impact of size, popularity, and sensitivity to cache space on cache performance, we express the weight CDW_i of content domain i as:

$$CDW_i = \varphi_1 * N_i * (\varphi_2 * P_i + \varphi_3 * S_i) \qquad i \in CD \tag{1}$$

In Expression (1), N_i represents the content size contained in content domain i, P_i represents the average popularity of content in content domain i, S_i is the average sensitivity of content in content domain i to the size of the cache, and CD is a collection of content domains in the network. φ_1,φ_2 , etc. are the weights of each factor, which can be adjusted according to the actual situation or analysis requirements. The size of the content in the content server can be easily obtained from the content provider. Nodes in ICN can handle transport layer protocols, so we switch from different application types to preset values of corresponding content size sensitivity by predefined lookups and bring them to the transport layer along with content popularity. Sensitivity values for different types of applications were determined from previous studies [36,37]. Content popularity can be collected and saved by individual nodes, then analyzed and calculated offline on additional servers. In many scenarios, content sensitivity to cache space and popularity can be difficult to obtain. In order to make our scheme more general and easy to implement, we simplify the weight CDW_i of content domain i as:

$$CDW_i = N_i \qquad i \in CD \tag{2}$$

In this case, the calculation of content domain weights only considers the size differences of the content contained within the content domain, and the popularity and sensitivity to the cache size of the content within each content domain are the same by default. This will increase the evaluation error of node importance and reduce the system performance. At the same time, it can make the algorithm easier to implement.

5.2. Data Path Weight

Data paths are the division of content domains from the user's perspective, so the weight of data paths is mainly determined by the forwarding process of requests sent by the user. First, the number of users will undoubtedly have an important impact on the number of requests on the path. Data paths connecting multiple users contain a large number of diverse requests. Such nodes should be assigned more cache sizes, so they weigh higher than paths with fewer connected users. Second, we consider the impact of user request habits. We assign greater weights to data paths whose request preferences are consistent with the content domain to which the current data path belongs. Since the request preference of a data path is inconsistent with the content of the current path, which means that the content the user wants is not in that content domain, we will allocate space to it in the content domain corresponding to the content. Finally, similar to content domain weight calculations, the sensitivity of the type of user requesting content to the size of the cache can also have a significant impact on cache performance in data paths.

Combining the number of users, request preferences, and the sensitivity of the requested content to the size of the cache space, we express the weight DPW_j of the data path j as:

$$DPW_j = \alpha_1 * R_j * (\alpha_2 * PR_j + \alpha_3 * S_j) \qquad j \in DP \tag{3}$$

In Expression (3), R_j represents the number of users connected by the data path j. PR_j indicates that the request preferences of the users connected by path j are consistent with the content domain to which the path belongs. And DP is a collection of data paths in a content domain. The sensitivity of the type of content requested by the user to the size of the cache is represented by S_j. α_1, α_2, etc. are the weights of each factor, similar to that of content domains, and can be adjusted based on actual conditions or analysis requirements. The sensitivity of content to cache space can be obtained by predefined table lookup. User's request preference can be regarded as the popularity of content among corresponding users, which is collected and saved by each node, and then analyzed and calculated on additional servers. In many scenarios, it is very difficult to obtain the sensitivity of content to cache space and user request preference, but it is easy to obtain the number of users. In order to make our scheme more general and easy to implement, the weight DPW_j of the data path can be simplified to:

$$DPW_j = R_j \qquad j \in DP \tag{4}$$

In this case, the calculation of the weight of the data path ignores the user's request preferences and the sensitivity of the requested content to the cache size, only considering the effect of the number of users. This will increase the evaluation error of node importance and reduce the system performance. At the same time, it can make the algorithm easier to implement.

5.3. Node Weight

Within a data path, the relationship between nodes is simple, and the factors affecting cache performance are not complex. We mainly consider the node's location in the path, that is, the distance from the node to the user or content server, and express the weight NW_m of node m as:

$$NW_m = D_m^{\beta} \qquad m \in N \tag{5}$$

In Expression (5), D_m represents the distance from the node to the content server. N is a collection of nodes in a data path. When $\beta > 0$, node weight is positively correlated with D_m, and node weight closer to the user is larger. When $\beta < 0$, node weight is negatively correlated with D_m, and node weight near content server is higher. When $\beta = 0$, the node weights are not related to D_m, and the weights of each node on a data path are the same. It is generally assumed that nodes that are closer to the content server should allocate more cache sizes, that is, the weight of the node should be higher. However, studies have shown that in some cases, allocating more cache sizes to nodes near the edge of the user

can lead to better cache performance and that nodes closer to the user should be allocated more cache sizes. Therefore, the β should be adjusted to the specific network structure or analysis requirements.

5.4. Global Weight

Based on content domain weights, data path weights, and node weights, we calculate the global weights of each node in the network. Since different domains contain the same node, and so do different paths, we need to analyze both cases specifically.

For a node that belongs to multiple different content domains at the same time. If a node belongs to a content domain, the content of that content domain needs to be cached. Because different content domains contain different content, nodes need to store the content contained in each content domain to which they belong. So we overlay the weights that nodes are assigned from different content domains to get the following NG_n:

$$NG_n = \sum_{a=1}^{CDC} NCD_a \tag{6}$$

In Expression (6), NCD_a is the value of the weight that Node n is assigned in content domain a. CDC is a collection of all content domains that contain the node.Within content domain a, the calculation of the weight value NCD_a assigned to a node takes into account that the node belongs to multiple different data paths at the same time.

For a node that belongs to multiple different content domains at the same time. From the content storage perspective, the content that needs to be cached is the same between different data paths in the same content domain, so it seems that the maximum value of node weight on each data path can satisfy the requirement of content caching. For example, a node should allocate 3 cache units in Path 1 and 5 cache units in Path 2. Since the content is the same, allocating 5 cache units to a node can meet the requirements of Path 1 as well as Path 2. However, from the perspective of traffic load, a node belonging to different data paths has a high traffic load, so we choose to add the weights allocated by the nodes in different data paths as the weights of the nodes in the content domain, as shown below, which can also avoid the complex calculation caused by maximizing:

$$NCD_a = \sum_{b=1}^{DPC_a} NDP_{ab} \tag{7}$$

In Expression (7), NDP_{ab} is the value of the weight that the node is assigned on data path b in content domain a. DPC_a is a collection of all data paths that contain this node in content domain a. With Expression (7), we can get the weight of a node in the content domain based on the weight that the node is assigned on the path. In combination with the Expression (6), we can further calculate the global weights of nodes based on their weights in the content domain.

$$NG_n = \sum_{a=1}^{CDC} \sum_{b=1}^{DPC_a} NDP_{ab} \tag{8}$$

In Expression (8), CDC represents the set of all content domains containing that node, DPC_a is the set of all data paths containing that node in content domain a, and NDP_{ab} is the weight assigned to that node on the data path b in the content domain a. The size of cache space allocated to a node is proportional to its global weight.

6. Results

Network hierarchy and weights based on the content distribution process have been clarified. The total cache budget is allocated to each node in proportion to its global weight. In this section, we describe the experimental process. First, we introduce the experimental parameters, then we introduce and analyze the experimental results.

6.1. Experimental Setup

A classical ICN network topology GARR (Italian academic network) is selected for simulation. The experiment is completed on a Python-based ICN simulation platform.

Table 2 shows the basic parameters for our simulation. There are 104 nodes in the network, 13 of which are source nodes connected to the content server, 20 of which are edge nodes connected to users, and the others are intermediate nodes. All content servers store 3×10^5 content that satisfies the Zipf distribution expressed as $p_i = \frac{i^{-\alpha}}{\sum_{k=1}^{|C|} k^{-\alpha}}$. The popularity of content i is determined by both the parameter α and the content set size C. Of the 13 content servers, we randomly selected two as large servers with a weight of 80; one as a medium server with a weight of 25; and the other as small servers with weights ranging from 1 to 10. We place the contents in 13 content servers according to the above weights, and use Expression (2) to calculate the weights of each content domain, Expression (3) to calculate the weights of each data path, then use Expression (5)–(8) to calculate the global weights of each node, and finally allocate cache space for each node based on its weights. Requests from all users are generated with the same probability. To conform to the normal operation of the actual network, we first generated 3×10^5 requests to preheat the system so that the empty cache of each node in the network stores hot network content. We then generated 6×10^5 more requests to collect data. All the requests are independent. From the previous research [38,39], the Zipf distribution parameter α on web content ranges from about 0.6 to 0.9, so we set α based on this range. We use the ratio of total cache size to total content size to represent the total cache budget rather than its absolute value, which represents the relative amount of cache space to the total content. We use 0.01 to 0.11 as the value range for the total cache budget. We recorded the cache hit ratio of each cache node in each experiment and calculated the average cache hit ratio, which is the clearest indicator of cache system performance. Average latency is also measured in each experiment to show how the caching system improves the user experience. In addition, we recorded the amount of content delivered per unit time of each content server, which can reflect the traffic load of the content server.

Table 2. Experiment parameters.

Parameters	Value
Topology structure	GARR
Content placement	LCD
Replacement policy	LRU and LFU
Total number of network nodes	104
Content space size	3×10^5
Requests number for system warm-up	6×10^5
Requests number for system data collection	6×10^5
Requests per second	12
Range of total cache space	0.01–0.11
Range of Zipf parameter α	0.6–0.9
Experiment run time for each scenario	10

In cache allocation scenarios, LCE (Leave Copy Everywhere) cache content placement strategies are often used as default strategies in ICNs. This scheme is simple and easy, but it causes a lot of unnecessary redundancy in the cache on the network, so LCD (Leave Copy Down) [40] content placement strategy is often used instead of LCE to reduce redundancy in content copy. We experimented with the LCD placement strategy and compared the effects of different cache replacement policies on cache size allocation schemes through LRU (Least Recently Used) [41] and LFU (Least Frequently Used). LRU will replace content that has not been accessed for the longest time in the most recent period with new content, while LFU will replace content that has been accessed least in the most recent period. We use three traditional cache allocation schemes, including homogeneous cache allocation scheme (represented as UNIFORM) and two heterogeneous cache allocation schemes, degree centrality based cache allocation (represented as DEGREE) and betweenness centrality based cache allocation (represented as BETWEENNESS), for comparison with the hierarchi-

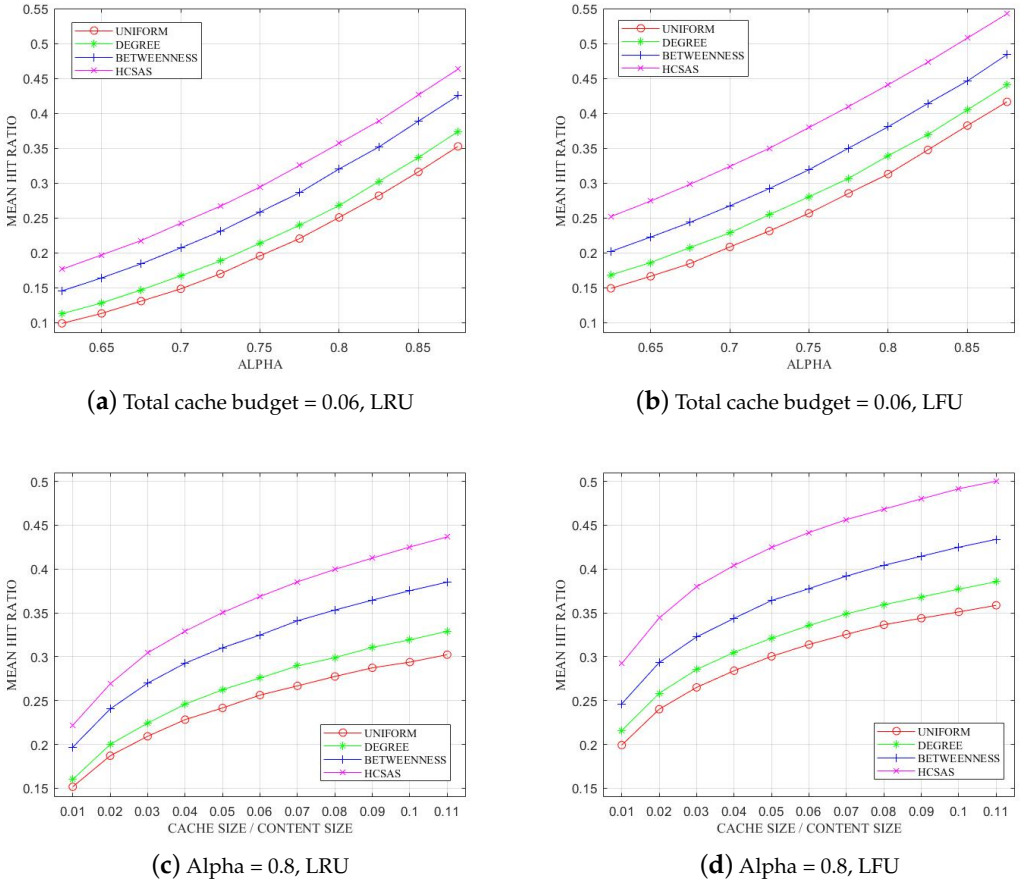
cal cache size allocation scheme (represented as HCSAS) presented in this paper. Uniform evenly allocates the total cache budget to all cache nodes in the network. DEGREE and BETWEENNESS respectively allocate the cache space according to the degree centrality and the betweenness centrality of the cache nodes. We compare the proposed scheme with these methods in two ways, including the Zipf distribution parameter alpha and the total cache budget.

6.2. Cache Hit Ratio

Node cache hit ratio refers to the proportion of requests for content that has been cached by a node to the total number of requests arriving at a node. We averaged the cache hit ratio for all nodes in the network to reflect the overall performance of the cache system.

In Figure 7a,b, the average cache hit ratio for each cache size allocation scheme increases with the increase of the Zipf distribution parameter alpha. This is because the increase in Alpha results in more centralized distribution of hot spots, which are more likely to be cached in nodes. Among all the test values, the scheme presented in this paper maintains its advantages.

In Figure 7c,d, the increase in the total cache size budget enables nodes to cache more content, so the average cache hit ratio increases with the increase in the cache budget. However, as the total budget increases, the ratio of cache hits increases slowly. Cache hit ratios differ significantly between different scenarios. The scheme proposed in this paper only needs a 0.05 total cache budget to achieve the performance when the BETWEENNESS scheme total cache budget is 0.11, saving more than half of the cache budget, and saving more cache budget than UNIFORM and DEGREE.



(**a**) Total cache budget = 0.06, LRU

(**b**) Total cache budget = 0.06, LFU

(**c**) Alpha = 0.8, LRU

(**d**) Alpha = 0.8, LFU

Figure 7. Performance of Cache Hit Ratio.

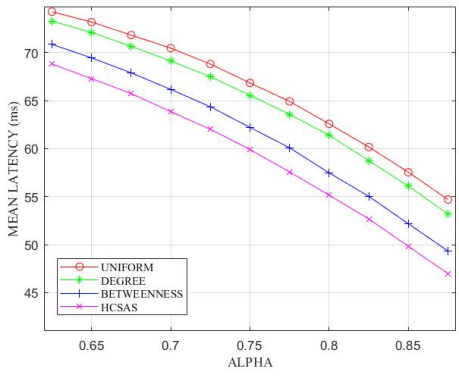
The LFU with awareness of request frequency makes each cache size allocation scheme more efficient. HCSAS performs best in different cache schemes, which significantly improves the overall cache hit ratio of the system. Since the content dissemination-based network hierarchy partitioning in the HCSAS scheme avoids errors caused by mismatches between the network topology and the content distribution process, the HCSAS scheme can accurately allocate more cache space to more important nodes who need it in the content distribution process, thus improving the cache hit ratio of the system. Compared with the cache allocation schemes such as BETWEENNESS, HCSAS can more effectively utilize the limited total cache budget and achieve a certain cache hit ratio HCSAS scheme requires less total cache budget.

6.3. Request Latency

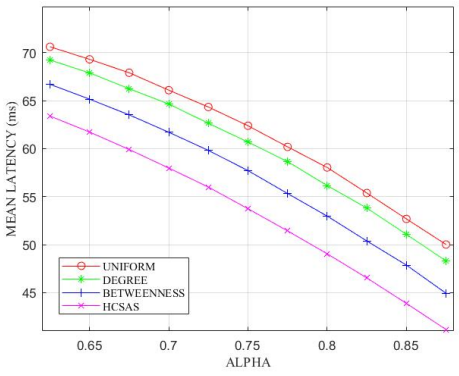
We recorded the time taken for the requested content to reach the user and averaged it as the system's request latency to analyze the performance of different cache size allocation schemes from the user's perspective.

As alpha increases in Figure 8a,b, request latency decreases for all cache size allocation schemes. For larger alphas, user-generated requests are focused on highly popular content that is often cached by nodes. In this case, requests can respond on intermediate or edge nodes closer to the user than on the content server, which reduces transmission latency. And HCSAS has advantages over other solutions.

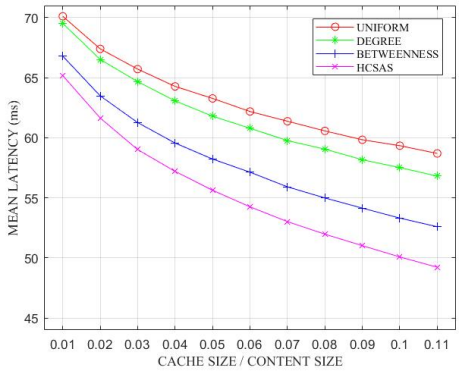
As the total budget for cache space in Figure 8c,d increases, request latency decreases at a slower and slower rate. As the total cache budget increases, so do the space allocated by nodes close to users, making it easier for users to get the requested content from nodes close to themselves. The request latency of HCSAS is significantly lower than that of other schemes, which only requires a total cache budget of 0.03 for a 40 ms latency and at least a total cache budget of 0.07 for a BETWEENNESS scheme.



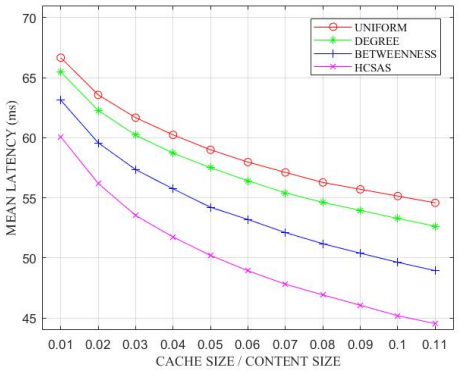
(**a**) Total cache budget = 0.06, LRU



(**b**) Total cache budget = 0.06, LFU



(**c**) Alpha = 0.8, LRU



(**d**) Alpha = 0.8, LFU

Figure 8. Performance of Request Latency.

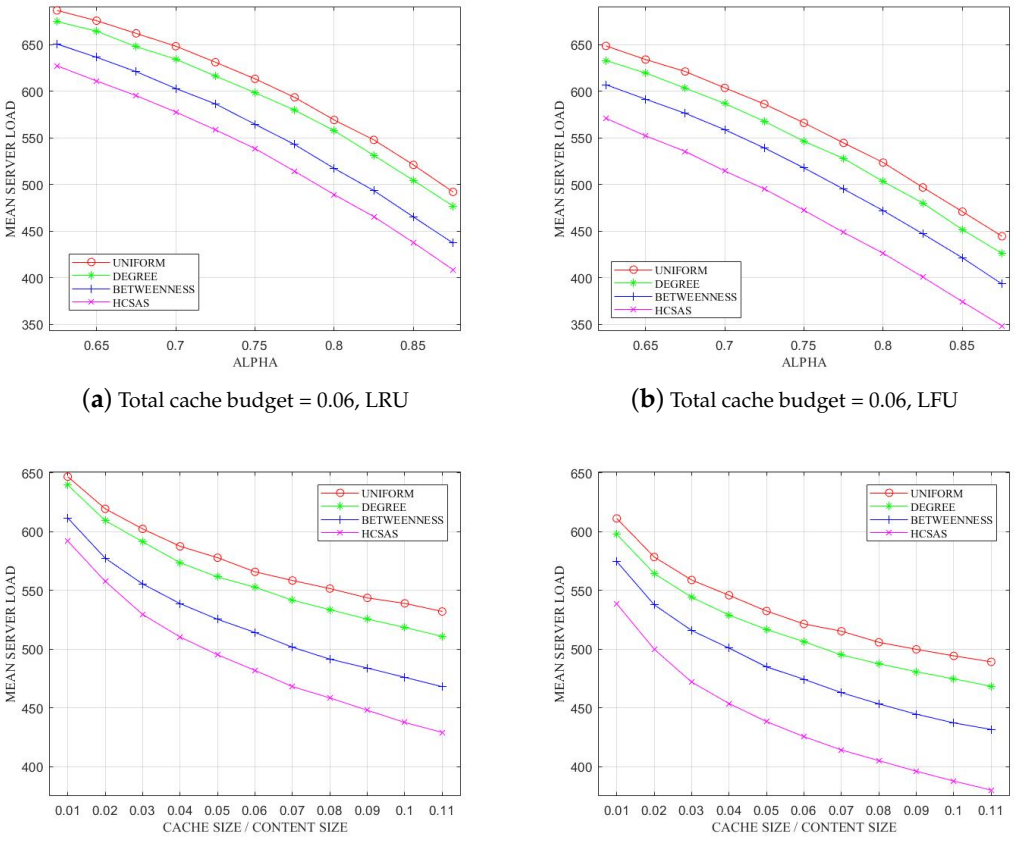
The HCSAS scheme still performs better than other cache size allocation schemes in reducing request latency.The intermediate nodes are closer to the users than the content servers, so the content transfer takes less time when the request is answered by an intermediate node. Also, the higher the cache hit ratio, the more likely the request will respond in the intermediate node, so the latency will be lower. Request latency is lower when LFU is used by each scheme. As a result, request latencies tend to be the opposite of cache hit ratio. Compared with the other three schemes, the HCSAS scheme has the highest cache hit ratio and therefore the lowest request latency given the cache budget.

6.4. Content Server Load

The average number of contents transferred per second by 13 content servers was recorded in each experiment. The average value of these data indicates that the load on the network content server is used to analyze cache system performance from the content provider's perspective.

From Figure 9a,b, the content server load from each cache allocation scheme decreases as the Zipf distribution parameter Alpha increases. From the previous analysis, an increase in Alpha makes hot content more likely to be requested and cached, and a large number of requests are hit at the nodes, which reduces the load on the content server. HCSAS performs best in different Alphas.

From Figure 9c,d, the load on the content server decreases as the total cache budget increases. With large cache capacity, the load on the content server decreases more and more slowly. HCSAS has obvious advantages over other schemes in different total cache budgets. To reach the state where the content server transmits an average of 300 content per second, the BC scheme requires a total cache capacity of 0.11, while the HCSAS only requires a total cache capacity of 0.05, which saves significantly the cache space.



(**a**) Total cache budget = 0.06, LRU

(**b**) Total cache budget = 0.06, LFU

(**c**) Alpha = 0.8, LRU

(**d**) Alpha = 0.8, LFU

Figure 9. Performance of Content Server Load.

From the content provider's perspective, the HCSAS scheme is more effective in reducing the load on the content server. The more requests hit the node, the fewer requests reach the server and the lower the server load. Similar to request latency, content server load is inversely proportional to the cache hit ratio. In all schemes, the load on the content server in LFU is lower than that in LRU. In Alpha scenarios, HCSAS can achieve the same performance with less than half of the BC cache budget.

7. Discussion

7.1. Content Dissemination

In this article, based on the content dissemination, the network is divided into three levels: content domain, data path, and node. In the process of hierarchy, we mainly consider two processes: content from a server to user and request from users to server. The actual content distribution process is very complex. For the cache size allocation to fit the content distribution process more closely, some caching system models are needed to analyze the content distribution process in more detail.

7.2. Cache Size Validity

The simulation results show that the cache performance, including cache hit ratio, request latency, etc., improves slowly when the total cache size is large. We follow the principle that important nodes in the network are allocated large cache space. However, even if the importance of nodes can be expressed very accurately, the optimal cache allocation may not be fully proportional to the importance. For example, nodes A and B have weights of 2 and 8, respectively. With a total cache of 10, allocating 4 and 6 cache space to two nodes may work better than allocating 2 and 8. To improve cache performance, some adjustments need to be made based on the validity of the size of the cache, which is based on the allocation of cache space by node importance.

7.3. Future Research

In the future, the content distribution process will be analyzed more accurately according to the actual situation of the network, and based on the existing work, the cache size allocation will be optimized from other aspects such as the validity of the cache size.

8. Conclusions

In this article, we solve the cache size allocation problem for a given total cache budget in an ICN network with an arbitrary graph structure. Based on the content distribution process, a hierarchical cache size allocation scheme is proposed, which is proportional to the importance of nodes. First, we analyze the interaction between the content distribution and caching and build a network model of arbitrary graph structure for analysis. Next, according to the network topology and content distribution process, we divide the network into three levels: content domain, data path, and node. Then, the weights of the three levels are calculated according to different influencing factors. Finally, the global weight calculation method for nodes is proposed, and the total cache budget is allocated proportionally to each node according to its global weight. The simulation results show that the proposed HCSAS scheme can achieve better cache performance. From an ISP perspective, HCSAS can achieve higher cache hit ratios under a given total cache budget. From the user's point of view, the request latency for the HCSAS scheme is lower. Lower content server load is achieved in HCSAS from a content provider perspective.

Author Contributions: Conceptualization, H.L. and R.H.; methodology, H.L. and R.H.; software, H.L.; writing—original draft preparation, H.L.; writing—review and editing, H.L. and R.H.; supervision, R.H.; project administration, R.H.; funding acquisition, R.H. 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).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: We would like to express our gratitude to Jinlin Wang, Rui Han, Yuanhang Li, and Younan Lu for their meaningful support for this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Saxena, D.; Raychoudhury, V.; Suri, N.; Becker, C.; Cao, J. Named Data Networking: A Survey. *Comput. Sci. Rev.* **2016**, *19*, 15–55. [CrossRef]
2. Carofiglio, G.; Gehlen, V.; Perino, D. Experimental Evaluation of Memory Management in Content-Centric Networking. In Proceedings of the 2011 IEEE International Conference on Communications (ICC), Kyoto, Japan, 5–9 June 2011; pp. 1–6.
3. Ahlgren, B.; D'Ambrosio, M.; Marchisio, M.; Marsh, I.; Dannewitz, C.; Ohlman, B.; Pentikousis, K.; Strandberg, O.; Rembarz, R.; Vercellone, V. Design Considerations for a Network of Information. In Proceedings of the 2008 ACM CoNEXT Conference, Madrid, Spain, 9–12 December 2008; Association for Computing Machinery: York, NY, USA, 2008; pp. 1–6.
4. 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; Association for Computing Machinery: New York, NY, USA, 2007; pp. 181–192.
5. Lagutin, D.; Visala, K.; Tarkoma, S. Publish/Subscribe for Internet: PSIRP Perspective. *Future Internet Assem.* **2010**, *84*, 75–84.
6. Dannewitz, C.; Kutscher, D.; Ohlman, B.; Farrell, S.; Ahlgren, B.; Karl, H. Network of Information (Netinf)—An Information-Centric Networking Architecture. *Comput. Commun.* **2013**, *36*, 721–735. [CrossRef]
7. Sharma, A.; Tie, X.; Uppal, H.; Venkataramani, A.; Westbrook, D.; Yadav, A. A Global Name Service for a Highly Mobile Internetwork. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 247–258. [CrossRef]
8. Venkataramani, A.; Kurose, J.F.; Raychaudhuri, D.; Nagaraja, K.; Mao, M.; Banerjee, S. Mobilityfirst: A Mobility-Centric and Trustworthy Internet Architecture. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 74–80. [CrossRef]
9. 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, Rome, Italy, 1–4 December 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 1–12.
10. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named Data Networking. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
11. Wang, J.; Cheng, G.; You, J.; Sun, P. SEANet: Architecture and Technologies of an On-site, Elastic, Autonomous Network. *J. Netw. New Media* **2020**, *6*, 1–8.
12. Afanasyev, A.; Burke, J.; Refaei, T.; Wang, L.; Zhang, B.; Zhang, L. A Brief Introduction to Named Data Networking. In Proceedings of the MILCOM 2018—2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; pp. 1–6.
13. Agyapong, P.K.; Sirbu, M. Economic Incentives in Information-Centric Networking: Implications for Protocol Design and Public Policy. *IEEE Commun. Mag.* **2012**, *50*, 18–26. [CrossRef]
14. Passarella, A. A Survey on Content-Centric Technologies for the Current Internet: CDN and P2P Solutions. *Comput. Commun.* **2012**, *35*, 1–32. [CrossRef]
15. Zhang, G.; Tang, M.; Cheng, S.; Zhang, G.; Song, H.; Cao, J.; Yang, J. P2P Traffic Optimization. *Sci. China Inf. Sci.* **2012**, *55*, 1475–1492. [CrossRef]
16. Wierzbicki, A.; Leibowitz, N.; Ripeanu, M.; Wozniak, R. Cache Replacement Policies Revisited: The Case of P2P Traffic. In Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, 2004 (CCGrid 2004), Chicago, IL, USA, 19–22 April 2004; pp. 182–189.
17. Hefeeda, M.; Saleh, O. Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems. *IEEE/ACM Trans. Netw.* **2008**, *16*, 1447–1460. [CrossRef]
18. Arianfar, S.; Nikander, P.; Ott, J. Packet-Level Caching for Information-Centric Networking. In Proceedings of the Re-Architecting the Internet Workshop, Philadelphia, PA, USA, 30–31 November 2010; Association for Computing Machinery: New York, NY, USA, 2010; pp. 1–16.
19. Arianfar, S.; Nikander, P.; Ott, J. On Content-Centric Router Design and Implications. In Proceedings of the Re-Architecting the Internet Workshop, Philadelphia, PA, USA, 30–31 November 2010; Association for Computing Machinery: New York, NY, USA, 2010; pp. 1–6.
20. Fang, C.; Yu, F.; Huang, T.; Liu, J.; Liu, Y. A survey of energy-efficient caching in information-centric networking. *IEEE Commun. Mag.* **2014**, *52*, 122–129 [CrossRef]
21. Muscariello, L.; Carofiglio, G.; Gallo, M. Bandwidth and Storage Sharing Performance in Information Centric Networking. In Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking, Toronto, ON, Canada, 15–19 August 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 26–31.

22. Zhang, M.; Luo, H.; Zhang, H. A Survey of Caching Mechanisms in Information-Centric Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1473–1499. [CrossRef]
23. Kim, Y.; Yeom, I. Performance Analysis of In-Network Caching for Content-Centric Networking. *Comput. Netw.* **2013**, *57*, 2465–2482. [CrossRef]
24. Rossini, G.; Rossi, D. A Dive into the Caching Performance of Content Centric Networking. In Proceedings of the 2012 IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, Spain, 17–19 September 2012; pp. 105–109.
25. 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.
26. 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]
27. Cui, X.; Liu, J.; Huang, T.; Chen, J.; Liu, Y. A Novel Metric for Cache Size Allocation Scheme in Content Centric Networking. In Proceedings of the National Doctoral Academic Forum on Information and Communications Technology 2013, Beijing, China, 21–23 August 2013.
28. Wang, Y.; Li, Z.; Tyson, G.; Uhlig, S.; Xie, G. Optimal Cache Allocation for Content-Centric Networking. In Proceedings of the 2013 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 7–10 October 2013; pp. 1–10.
29. Krishnan, P.; Raz, D.; Shavitt, Y. The Cache Location Problem. *IEEE/ACM Trans. Netw.* **2000**, *8*, 568–582. [CrossRef]
30. Chauhan, S.; Girvan, M.; Ott, E. Spectral Properties of Networks with Community Structure. *Phys. Rev. E* **2009**, *80*, 056114. [CrossRef]
31. Mangili, M.; Martignon, F.; Capone, A. Optimal Design of Information Centric Networks. *Comput. Netw.* **2015**, *91*, 638–653. [CrossRef]
32. 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.
33. Wang, Y.; Li, Z.; Tyson, G.; Uhlig, S.; Xie, G. Design and evaluation of the optimal cache allocation for content-centric networking. *IEEE Trans. Comput.* **2015**, *65*, 95–107. [CrossRef]
34. 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]
35. Jin, D.; Lv, J. Optimal Heterogeneous Cache Allocation Mechanism In Information-Centric Networking. *Int. J. Innov. Comput. Inf. Control* **2021**, *17*, 193–203.
36. Lu, Y,; Abdelzaher, TF.; Saxena, A. Design, implementation, and evaluation of differentiated caching services. *IEEE Trans. Parallel Distrib. Syst.* **2004**, *15*, 440–452.
37. Fricker, C.; Robert, P.; Roberts, J.; Sbihi, N. Impact of traffic mix on caching performance in a content-centric network. In Proceedings of the IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 March 2012; pp. 310–315.
38. Breslau, L.; Cao, P.; Fan, L.; Phillips, G.; Shenker, S. Web caching and Zipf-like distributions: Evidence and implications. In Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, 21–25 March 1999; pp. 126–134.
39. 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.
40. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634. [CrossRef]
41. 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.