Service Prioritization in Information Centric Networking With Heterogeneous Content Providers

Jin Qin, *Graduate Student Member, IEEE*, Kaiping Xue, *Senior Member, IEEE*, Jian Li, *Member, IEEE*, Qibin Sun, *Fellow, IEEE*, and Jun Lu

***Abstract*—Service prioritization brings reasonable allocation of network resources and improves the overall quality of experience (QoE) of users, but it has not been thoroughly investigated in information centric networking (ICN). Existing works lack adaptability and they cannot ensure specific content provider (CP) get well caching service which is one of the most important functions in ICN. In this paper, we firstly propose a service prioritization scheme to flexibly provide different caching services for heterogeneous CPs to improve the overall network efficiency. The main idea is to allocate dedicated cache space for paying CPs and provide prioritized caching service for them, while normal CPs only enjoy the normal caching service. The scheme can be divided into two phases. First, we select a group of nodes with higher importance as core nodes based on network topology, and pair each edge node to a core node following the two-sided many-to-one matching algorithm. Second, we dynamically allocate and manage the dedicated cache space for core nodes. We model the allocation of dedicated cache space and convert it into a convex optimization problem to solve. After that, a practical caching strategy and system design are implemented in the ndnSIM simulator. Finally, we evaluate our scheme and conduct comparative experiments with the most representative work diff-caching, simulation results show that our scheme outperform it in terms of both delay and cache hit ratio.**

***Index Terms*—Information centric networking, named data networking, service prioritization, cache space allocation.**

I. Introduction

WITH the development of network technologies and rapid growth of users' quality demands, information content services represented by online videos and social networks have dominated Internet services. According to the Cisco Visual Networking Index [1], IP video traffic will be 82 percent of all IP traffic by 2022, up by 75 percent in 2017. Traditional TCP/IP network is a host-centric communication architecture, and when faced with these new network applications, it often encounters challenges such as redundant transmission, multi-path transmission, and user mobility. Information centric networking (ICN) [2] is an emerging network paradigm that decouples content from its storage location by providing one or more copies and distributing them across the network. Contents can be cached on any device at any location to answer the same requests received in the future, which expects to reduce redundant transmissions in the network.

Manuscript received February 24, 2021; revised July 12, 2021; accepted August 9, 2021. Date of publication August 16, 2021; date of current version December 9, 2021. This work is supported in part by the National Natural Science Foundation of China under Grants No. 61972371, and No. U19B2033, the Peng Cheng Laboratory Project under Grant PCL2021A02 and Youth Innovation Promotion Association Chinese Academy of Sciences (CAS) under Grant No. Y202093. The associate editor coordinating the review of this article and approving it for publication was A. Mourad. *(Corresponding author: Kaiping Xue.)*

Jin Qin, Jian Li, and Qibin Sun are with the School of Cyber Security, University of Science and Technology of China, Hefei 230027, Anhui, China.

Kaiping Xue is with the School of Cyber Security, University of Science and Technology of China, Hefei 230027, Anhui, China, and also with the Department of New Networks, Peng Cheng Laboratory, Shenzhen 518055, Guangdong, China (e-mail: kpxue@ustc.edu.cn).

Jun Lu is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, Anhui, China.

Digital Object Identifier 10.1109/TNSM.2021.3105198

In recent years, a variety of ICN architectures have been proposed, such as NDN [3], DONA [4], and NetInf [5]. Named data networking (NDN) is the most promising implementation of ICN which follows the content centric philosophy. Content acquisition in NDN is based on an user-driven publish/subscribe model, whereby users' requests may be satisfied by any network device. Interest packets and data packets are the two basic types of packets in NDN and they have an one-to-one correspondence. Due to the content-centric data acquisition and name-based forwarding control, data transmission in NDN is quite different from that in traditional TCP/IP networks. In NDN, receivers can get contents from the origin server or from any intermediate node that has the cached content. And content requests can be dynamically forwarded in multi-path because of the separation of routing and forwarding in NDN.

Compared with traditional networks, ICN brings multiple advantages, such as in-network caching, redundant transmission reduction, active security, and natural mobility support. Nevertheless, the limited resources in ICN cannot guarantee that all users receive satisfied services, so it is necessary to optimally allocate resources to improve the overall quality of experience (QoE) of users. Existing works mainly focus on some research problems such as traffic management and control [6], [7], effective utilization of network resources [8], [9] such as computing, cache and bandwidth, and differentiated processing of user requests [10]–[12]. The last category, i.e., differentiated processing of user requests, aims to provide service prioritization for specific users in ICN.

Service prioritization refers to providing different operations for different data packets or streams in network. The service prioritization framework in IP networks [13]–[15] has been proposed and standardized for a long time, which is mainly based on the ideas of traffic aggregation and classification management. When data traffic enters the network, it is classified or shaped in an edge router and the classification result is marked by a label. Each core router defines a specific forwarding operation for each type [16], [17]. Under this framework, various types of data traffic can be processed with different priority levels so that requests with strict latency or bandwidth requirements can be preferentially satisfied, thereby improving the overall QoE of users. However, the research on service prioritization in ICN has attracted a few attentions. Among the limited number of related works, the most representative one is proposed by Kim *et al.* [11], in which an IP-like diff-serv model in NDN is proposed to jointly consider both caching and forwarding strategies. But the existing works don't consider the optimal network resources allocation and their implementations lack adaptability as they cannot be adjusted flexibly according to the changes in network conditions. Besides, these works mainly provide differentiated processing for different kinds of contents according to the application types they belong to, and cannot ensure that the contents from specific CPs be served in a prioritized way. Therefore, we put forward a new idea to flexibly provide differentiated services for contents from heterogeneous CPs. We allow CPs pay to get prioritized services and thus ensuring their contents can be better served when compared with the contents from normal CPs, which can improve their users' overall experience of data acquisition effectively.

More specifically, we allocate dedicated cache space for paying CPs and provide them with prioritized caching service. In-network caching is one of the most important features in ICN. Content can be cached on any node in the network, an when the node subsequently receives a request for the same content, it can directly return the local cached copy without forwarding the request to the corresponding content server. Through caching strategy design [8], [9] and caching node placement [18], the cache hit ratio, the overall utilization of network resources and the users' content acquisition experience can all be greatly improved. However, network resources in ICN are still limited and caching is performed at the granularity of data chunks. Whether a content can be locally cached or not usually depends on the comparison with other contents in terms of their local request frequency or popularity. For all CPs, whether a content can be cached or not is equal in the caching policies. For a specific CP, the existing standard caching policies in ICN cannot guarantee that its contents can be cached with a greater probability compared with other contents, and in the worst case there is not even any copy of its specific content in the network. Theoretically, in addition to the normal cache, if some prioritized CPs can occupy an additional portion of the cache space in a core node, it's inevitable that more of these CPs' contents will be able to cached with a greater possibility compared with contents from other normal CPs. This can ultimately better satisfy their users' requests. Therefore, in our proposed scheme, prioritized CPs are allowed to pay to rent dedicated cache space, which are only used to cache the contents from these prioritized CPs, so that they can get the guaranteed and prioritized caching service. Such design cannot only ensure that the paying CPs get better caching service but also increase the income of Internet service providers (ISPs).

Based on the above considerations, we propose a new service prioritization scheme in ICN, in which the main idea is to allow paying CPs to rent dedicated cache space and then provide prioritized caching service for their contents at selected nodes. The design can be divided into two phases: Firstly, a group of core network nodes are selected to deploy dedicated cache space. Secondly, the optimal cache size is calculated to be allocated on each core node, and then we design the specific caching strategy for these core nodes. We further develop a practical system implementation in the ndnSIM [19] simulator. We conduct some experiments under different caching strategies and different network scenarios to verify the effectiveness and efficiency of our scheme. Then, we also conduct comparative experiments with the existing work [11] in different network conditions. Diff-caching [11] is one of the most representative woks, which comprehensively applies the diff-serv architecture in IP to NDN, and provides different caching services for users according to the priorities of their requests. Simulation results show that our scheme outperform diff-caching in terms of both latency and cache hit ratio. The main contributions of this paper can be briefly summarized as follows.

- We propose a service prioritization scheme in ICN to provide different caching services for heterogeneous CPs. Through the allocation and management of dedicated cache space, we provide prioritized caching service for paying CPs while other normal CPs only enjoy normal caching service. We further model the main problem to decide how much dedicated cache space to allocate on which nodes, and it is solved in two phases.
- On which nodes to deploy the dedicated cache, we propose a optimized core node selection and matching scheme, in which a group of nodes with higher importance are selected as core nodes based on the network topology, and each edge node is matched to a specific core node following the two-sided many-to-one matching algorithm.
- We propose the dedicated cache space allocation and management scheme for core nodes. The allocation of dedicated cache space is formulated and converted into a convex problem which is solved with interior-point method. And a practical caching strategy and system implementation are proposed to manage the cache space.
- We realize our proposed service prioritization scheme in the ndnSIM simulator and evaluate it in different network scenarios. We also conduct comparative experiments with the existing work, i.e., diff-caching [11], simulation results verify the effectiveness and efficiency of our scheme.

The rest of this paper is organized as follows: the background and related works are introduced in Section II. The problem formulation and our proposed service prioritization schemes are briefly described in Section III. The details of node selection and matching is presented in Section IV. The cache space allocation and management scheme is given in Section V, which is followed by the system implementation in Section VI and the performance evaluation in Section VII. Finally, we conclude the paper in Section VIII.

II. Related Work

As an emerging network paradigm, ICN has attracted more and more attentions from both academia and industry. In order to improve the overall QoE of users and network efficiency, many researches have been carried out from several aspects, e.g., traffic management and control [6], [7], efficient use of network resources [8], [9], and the optimization of user services. While the former two aspects have been studied well, only few attentions are paid to the optimization of user service in ICN. In traditional networks, it's an effective way to provide prioritized services on demand according to the different types of data packets or streams [13], [14].

Service prioritization has been extensively studied in traditional TCP/IP networks [15]–[17] which have shown that under certain network conditions, it is possible to improve overall network performance and users' QoE by providing differentiated services. These works can be classified into two categories, e.g., to provide prioritized services for different applications or provide prioritized services for different CPs. There are only a few related works in ICN [10]–[12], and they all mainly focus on how to provide suitable services for different applications. Huo *et al.* [10] proposed to distinguish different kinds of contents and dynamically allocate the cache size on-demand according to different contents' popularity, but the scheme needs to frequently count all users' requests and uses a centralized method to send the cache decision to each node, which will bring about huge communication overhead. In addition, this work mainly focuses on the statistics and real-time weight calculation of the content popularity of each application, and does not give a practical and feasible deployment scheme. As the most representative one, Kim *et al.* [11], [12] applied an IP-like diff-serv model in NDN, including the differentiated forwarding model "diff-forwarding" and the differentiated cache model "diff-caching". In the diff-forwarding model, forwarding service is classified into three levels and the priority of each content request is marked at edge node. At the edge nodes, the sending rate of content requests is controlled according to their priorities. At the intermediate nodes, it also provides more convenient forwarding services for high-priority content requests. For the diff-caching model, both the caching service and spaces are divided into four levels, and corresponding service is provided according to the priority of the user request. Although this scheme can effectively enhance the overall utilization of network resources, it also has some shortcomings. On the one hand, the implementation efficiency depends on the values of some parameters, such as the interest marking rate or the size of caching window, which cannot be flexibly adjusted according to network conditions. On the other hand, the scheme does not provide an optimization method for cache resource allocation and cannot ensure that the specific CPs receive guaranteed services.

Another kind of works on service prioritization aims to provide prioritized services for heterogeneous CPs, which allows specific CPs to occupy more network resources such as bandwidth or cache space though paying to ISPs. There also have been some similar works in traditional networks and some representative works in CDN (Content Delivery Network). In these schemes, some CPs deploy CDN nodes in different operating networks and pay ISPs to rent dedicated bandwidth. MPLS (Multi-Protocol Label Switching) [20], [21] and SDN (Software Defined Networking) technology [22] can be applied to ensure the dedicated bandwidth allocation. The leased bandwidth is used exclusively to transmit the contents from specific CPs, thereby accelerating the content distribution efficiency, that is, the ISP provides prioritized services to the paying CPs. Specifically, Xu and Xu [23] adopted the MPLS technology to provide leased lines with high bandwidth, low latency, and high reliability services. Google used SDN technology to build B4 [24] to form a private wide area network that connects to global Google data centers. Therefore, by referring to these works, we can also design a suitable scheme to improve the overall network service quality for those paying CPs in ICN.

As mentioned before, CPs can pay to obtain more network resources such as bandwidth or cache space at important nodes, then provide users with better data acquisition service and QoE. However, different from bandwidth leasing in CDN, each node in ICN has cache ability and user's content request may be satisfied at any node or server, so it is not feasible to rent dedicated bandwidth at some specific links. In-network cache is another one of the most representative and important functions in ICN. Through caching strategy design [8], [9] and the deployment of caching nodes [18], the overall utilization of cache resources and the users' contents acquisition experiences can both be greatly improved. However, most of the existing works focuses on designing caching strategies [25], [26] executed on single node or multiple nodes with cooperation, and still few attentions are paid to the deployment and allocation of cache space. Wang *et al.* [18] proposed an optimal cache space allocation scheme under limited cache resources to optimize the overall network performance, but they did not consider the competition among heterogeneous CPs. Therefore, in order to provide users with better content acquisition experience, it's a good and practicable choice to provide prioritized caching service for paying CPs through the deployment of dedicated cache space.

III. Problem Statement

A. System Model

The whole network in our scheme mainly consists of three kinds of components: CPs, routers and users. The system model can be described as follows. CPs represent content providers with servers, and users' requesting interests are regularly distributed on various contents. It is also assumed that a content is permanently stored by one of these content servers. Routers are cache-enabled in the network and the caching space in a router can be divided into normal cache space and dedicated cache space. Users send content request packets whose arrival is a random variable that follows the Poisson process while users' request preference follows the popularity distribution. Each ICN node contains three tables, i.e., Forwarding Information Base (FIB), Content Store (CS) and Pending Interest Table (PIT). FIB is used to forward user request (the interest packet) towards potential data sources. The FIB can be established through internal routing protocols, the most representative ones, e.g., OSPFN [27] and NLSR [28], respectively focus on the shortest paths and the

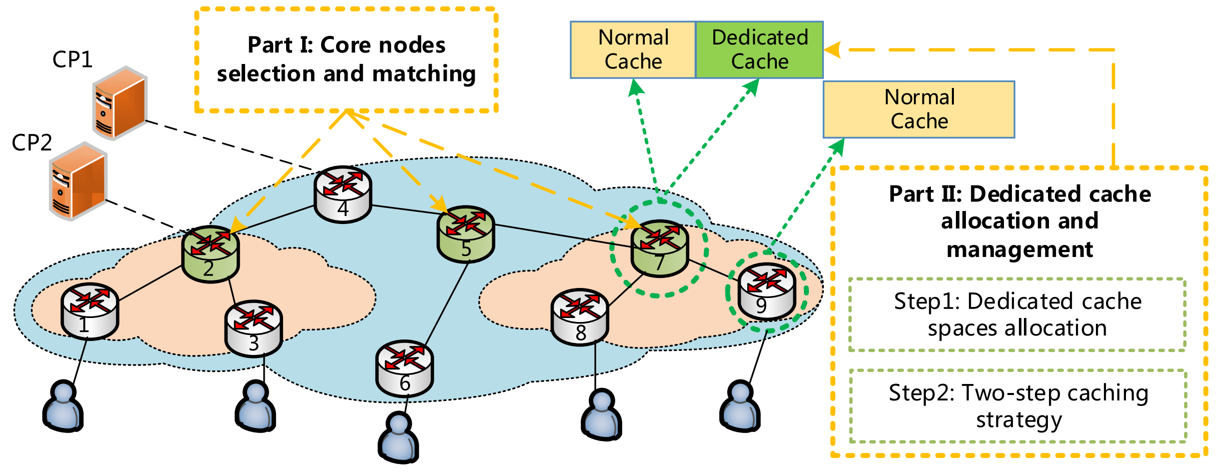

CP1
CP2
Part I: Core nodes selection and matching
Normal Cache
Dedicated Cache
Normal Cache
Part II: Dedicated cache allocation and management
Step1: Dedicated cache spaces allocation
Step2: Two-step caching strategy
1
2
3
4
5
6
7
8
9

Fig. 1. Problem Overview.

TABLE I
NOTIONS AND DEFINITIONS

SYMBOL	DESCRIPTION
$B_{(v)}$	Betweenness centrality of node v.
$C_{(v)}$	Closeness centrality of node v.
ω_v	Importance of node v.
EN	Set of edge nodes.
CN	Set of core nodes.
UM_EN	Set of unmatched edge nodes.
UM_CN	Set of unmatched core nodes.
PF_{EN}	Preference list of edge nodes.
PF_{CN}	Preference list of core nodes.
$u_{CN,m}()$	Utility function of core node m.
$u_{EN,n}()$	Utility function of edge node n.
Φ	The matching between EN and CN.
(cn_m, en_n)	The pair of edge node n and core node m.
C	The total budget of paying CP.
m_n	Dedicated cache space size on core node n.
p_n	Unit price of cache space size on core node n.
req_v	Users' requests for paying CP's contents at edge node v.
d_{v-cp_i}	Distance between node v and paying CP i.
d_{vn}	Distance between node v and node n.
$g(x)$	Probability density function of users' requests.
R_n	The revenue of dedicated cache space at core node n.
R_{total}	The total revenue of all core nodes' dedicated cache space.

link states. CS is used to record information about the locally cached contents under different strategies. And PIT keeps track of the traverse path of received interest packets so that the response data packets can be forwarded along the reverse path. All contents take chunk as the basic unit and each chunk has the same size in our scenario.

In this paper, we use an undirected graph $G = (V, E)$ to represent the network, where $V = (v_1, v_2, \ldots, v_n)$ refers to the set of nodes and $E = (e_1, e_2, \ldots, e_l)$ refers to the set of edges or links. Besides, $C_i = (c_{i1}, c_{i2}, \ldots, c_{iN})$ represents the set of all contents and $CP = (cp_1, cp_2, \ldots, cp_i)$ represents the set of content providers. For convenience, Table I describes the main notations we use in this paper.

B. Problem Overview

In this work, we aim to provide prioritized caching service for paying CPs through the deployment of dedicated cache space. The overall research problem is to solve how much cache space we need to deploy on which nodes to optimize overall network performance. As shown in Fig. 1, we can divide this problem into two parts:

- Firstly, we need to select a group of core nodes for deploying dedicated cache space, which will be introduced in details in Section IV. In our proposed scheme, we evaluate the importance of each node with the network topology information and then select a group of nodes with higher importance as core nodes, which are candidate cache deployment nodes. Then we further match edge nodes with core nodes, that is, for each edge node, choose a core node that is most likely to serve it in the subsequent process.
- Secondly, we focus on the allocation of dedicated cache space and the design of caching strategy on core nodes, which are described in Section V and Section VI, respectively. The allocation of dedicated cache space is formulated as a global optimization problem and we calculate the optimal cache size in each core node based on the matching results and user requests distribution. And we design a two-step caching strategy for core nodes with two pieces of independent cache space to achieve prioritized caching service for paying CPs.

The normal cache space is used for all contents going though this node and the dedicated cache space is only used to cache the passed contents from paying CPs. Accordingly, each core node provides two types of in-network caching services for heterogeneous CPs. Paying CPs enjoy the prioritized caching service, and their contents can leverage both the normal cache space and the dedicated cache space in each core node. Meanwhile, normal CPs can only enjoy the normal caching service and their contents can only use the normal cache space.

Specifically, in the first part, we evaluate the importance of nodes by their geographic locations in the network topology. When making pairs between edge nodes and core nodes, we model it as a many-to-one matching problem between two disjoint sets and solve the problem based on the well-known Gale Shapley [29] algorithm. In the second part, with the matching results and the user request information at each edge node, we can calculate the maximum cache revenue that an unit of dedicated cache space can bring. Based on the revenue and the constraints of the size of rentable cache space and the CP's budget, we convert the optimal dedicated cache allocation problem into a convex optimization problem with both equality and inequality constraints, and use the interior-point method to solve it. After that, we propose a practical system implementation and design the relevant protocols and caching strategy on core nodes. With the two-step caching strategy, core nodes provide prioritized caching service for the paying CPs. For each passed content, our scheme first determines whether it needs to be cached in the normal cache space based on the specific caching strategy of the normal caching service. If not, the scheme further judge whether it belongs to a paying CP or not. If so, the scheme decides whether the content needs to be cached in the dedicated cache space based on the caching strategy of the prioritized caching service, which is introduced in detail in Section VI.

IV. Core Nodes Selection and Matching

A. Overview

We aim to provide prioritized caching service by allowing paying CPs to rent parts of dedicated cache space at appropriate nodes. Nevertheless, it is unreasonable to deploy dedicated cache space at every node in the whole network due to the high cost and low efficiency. Therefore, we divide the problem into two parts to solve in sequence. First, we propose a method to evaluate nodes' importance in order to select a group of core nodes under a given network topology. Intuitively, these nodes should serve more potential network traffics and cover more edge nodes. Second, we propose a matching algorithm to find the pairing relationships between core nodes and edge nodes. It means that, for each edge node, find the core node that is most likely to serve it in the subsequent process.

B. Evaluation of Node's Importance

A node's importance is mainly determined by its location and we quantify this attribute from two aspects here. We first use betweenness centrality (BC) [30] parameters to quantify the potential traffics of nodes. The value can reflect the probability that the intermediate node receives the interest packet when the new interest enters the network, that is, the potential traffic. The larger the value is, the greater the probability of the new interest packet passing through node *n* will be, that is also the greater the potential traffics at this node is. The betweenness centrality of a node can be expressed as follows. $\sigma_{i,j}$ represents the number of shortest paths between the pair of *i* and *j*, and the variable $\sigma_{i,j}(v)$ represents the number of shortest paths passing through node *v*.

$$B_{(v)} = \sum_{i \neq j \neq v \in V} \frac{\sigma_{i,j}(v)}{\sigma_{i,j}}. \tag{1}$$

Then we use closeness centrality [31] to quantify the request delay of each node in the network. The closeness centrality indicates the overall distance between the intermediate node *v* and other nodes in the network, which reflects the potential delay when node *v* request content from other nodes. The calculation of closeness centrality is shown as

$$C_{(v)} = \frac{1}{\sum_{j \in V} d(v,j)}, \tag{2}$$

where *d*(*v*, *j*) represents the shortest path length from node *v* to other node *j*.

Jointly considering the values of node betweenness centrality and closeness centrality, the importance of each node in the network can be quantified as

$$\omega_v = a_1 \frac{B_{(v)}}{B} + a_2 \frac{C_{(v)}}{C}, \tag{3}$$

where $a_1 + a_2 = 1$, *B* represents the sum of betweenness centrality of all nodes in the network, and *C* represents the sum of closeness centrality. By calculating the importance of nodes, we select a group of core nodes. In our scheme, the selected criterion is that the importance of nodes is in the top 10% of all nodes and the weight parameters a_1 and a_2 are both set to 0.5. Then for each edge node, we find the core node that best matches it, that is, the core node that is most likely to satisfy the user requests come from this edge node.

C. Matching Between Core Nodes and Edge Nodes

Two sided matching is one of the most typical problems in matching theory, which can be seen as a pairing between two disjoint sets. According to the matching type, problems can be classified into one-to-one [32], many-to-one [33], and many-to-many [34] matching. In our scenario, we need to select a proper core node for each edge node. For each edge node, its associated core node is an element from the limited core nodes set. Such association problem can be regarded as a two-sided matching problem between the core nodes set and the edge nodes set. For each edge node, its corresponding core node is unique, while for each core node, there may be multiple edge nodes that will be served by it.

The pairing between core nodes and edge nodes can be regarded as a process in which the elements of two sets aim at matching with each other. Edge nodes and core nodes can be considered as two disjoint sets of selfish and rational players who aim to maximize their own interests. If edge node en_n is matched with core node cn_m, that is, en_n will more likely get contents from cn_m in the subsequent process, we say that en_n and cn_m form a *matching pair*. The matching is defined as

Definition 1: For the two disjoint sets of the core nodes $CN = \{cn_1, cn_2, \cdot, cn_M\}$ and the edge nodes $EN = \{en_1, en_2, \ldots, en_N\}$, a many-to-one two-sided matching Φ is a mapping from the set $CN \cup EN$ into the set of all subsets of $CN \cup EN$. For each $cn_m \in CN$ and $en_n \in EN$, there are:

1) $\Phi(cn_m) \subseteq EN, 0 \leq |\Phi(cn_m)| \leq k < N$;
2) $\Phi(en_n) \subseteq CN, 0 \leq |\Phi(en_n)| \leq 1$;
3) $\Phi(en_n) = \{cn_m\} \Leftrightarrow en_n \in \Phi(cn_m)$.

where *k* is the maximum number of edge nodes that the core node can match, and it is used to avoid too many edge nodes matching the same core node, which may cause potential link congestion and resource imbalance. For a matching game, each player in one set is always matched with one or more players in the other set. For the system model in our scheme, each core node can be assigned up to *k* edge nodes, but each edge node can choose only one core node. Besides, it should be noted that, the two players in a matching pair are symmetrical, which means if an edge node *A* is one of the matching results of core node *B*, then the core node *B* must also be the matching result of edge node *A*, and this relation is also valid in reverse.

The matching process and results are affected by the competitive relationship between players. We assume that each player in matching Φ has a preference for other players in another set. For example, the player of edge nodes holds a total list over $CN \cup \varnothing$. And similarly for core nodes, they can be represented as:

$$PF_{EN} = [PF_{EN}(1), \ldots, PF_{EN}(n), \ldots, PF_{EN}(N)],$$
$$PF_{CN} = [PF_{CN}(1), \ldots, PF_{CN}(m), \ldots, PF_{CN}(M)].$$

A player's preference list can be calculated by its corresponding utility function. For an edge node $en_n \in EN$ and its utility function $u_{EN,n}$, if $u_{EN,n}(m) > u_{EN,n}(v)$, we call the edge node en_n prefers core node cn_m to core node cn_v and it can be expressed as

$$cn_m \succ_{en_n} cn_v \Leftrightarrow u_{EN,n}(m) > u_{EN_n}(v), \tag{4}$$
$$u_{EN,n}(m) = min_{l \in L_{nm}} d_{nm,l}, \tag{5}$$

where L_{nm} the set of all feasible paths between edge node en_n and core node cn_m, $d_{nm,l}$ is the total length of path *l*.

Similarly, for a core node $cn_m \in CN$ and its utility function $u_{CN,m}$, if $u_{CN,m}(n) > u_{CN,m}(s)$, we call the core node cn_m prefers edge node en_n to edge node en_s, and it can be expressed as

$$en_n \succ_{cn_m} en_s \Leftrightarrow u_{CN,m}(n) > u_{CN,m}(s), \tag{6}$$
$$u_{CN,m}(n) = max\_usernum(n), \tag{7}$$

where $max\_usernum(n)$ is the maximum number of users who can access the network through edge node en_n.

D. Algorithm Designing and Its Analysis

Before proposing the matching algorithm, we first give the following definition.

Definition 2: For a matching Φ and pair (cn_m, en_n), if $cn_m \succ_{en_n} \Phi(en_n)$ and one of the following two conditions is true, then Φ is blocked by (cn_m, en_n). Namely, (cn_m, en_n) is a blocking pair.

1)$\exists en_s \in \Phi(cn_m), en_n \succ_{cn_m} en_s$,

2)$|\Phi(cn_m)| \leq k-1, \Phi(cn_m) \cup \{en_n\} \succ_{cn_m} \Phi(cn_m)$.

Based on the above definitions, we further propose the two sided many-to-one matching algorithm based on the well-known Gale-Shapley algorithm [29], which contains the two processes of initialization and matching: In the initialization process, each one in all edge nodes and core nodes calculate their preference lists based on the utility functions $u_{EN,n}(m)$ and $u_{CN,m}(n)$, respectively. The matching process is divided into two stages:

- In **stage 1**, each core node will send matching request to its current most preferred edge node according to its preference list. Each edge node accepts the request of its most preferred core node, so these two nodes are considered to have completed the pairing. This pair will be recorded in matching Φ and the two nodes will be removed from the unmatched nodes set UM_EN and UM_CN. The above steps will be repeated serval times until every core node forms a matching pair with one of its preferred edge node.
- In **stage 2**, in every round, each core node send matching requests to its at most *k* preferred edge nodes that have never rejected its requests before according to the preference list, where *k* is also the predefined maximum number of allowed matched edge nodes. Each edge node will choose and accept the request of its most preferred core node from all the requests it received in this round and its accepted request (if it has already matched with a core node) in last round, and then reject the requests of other core nodes. This stage will also repeat several times and the matching completes when no more core nodes send new requests. The matching process will converge to stability [35] after a certain number of iterations. The specific matching steps are shown in Algorithm 1.

Definition 3: A matching Φ is stable if it is not blocked by any individual or any pair.

Theorem 1: After limited iterations, the proposed algorithm converges to a stable matching Φ^*.

Proof: First, as it is shown in Algorithm 1, the two stages both have limited rounds. Stage 1 executes at most *M* rounds, and stage 2 repeats at most *N* rounds, where the actions in each round are limited. So the total iterations are limited. Second, with the detail in Algorithm 1, there are no $cn_m \in CN$ and $en_n \in EN$ that form a blocking pair in the final matching Φ^*. According to definition 3, we can know that Φ^* is stable. ■

Theorem 2: The stable matching Φ^* converged in proposed algorithm is Pareto optimal.

Proof: If Φ^* is not Pareto optimal, there must be a matching γ that satisfies condition 1)2) and one of condition3) or 4):

1) $\gamma(cn_m) \succeq_{cn_m} \Phi^*(cn_m), \forall cn_m \in CN$;
2) $\gamma(en_n) \succeq_{en_n} \Phi^*(en_n), \forall en_n \in EN$;
3) $\exists cn_m \in CN, \gamma(cn_m) \succ_{cn_m} \Phi^*(cn_m)$;
4) $\exists en_n \in EN, \gamma(en_n) \succ_{en_n} \Phi^*(en_n)$.

So if γ exists, there also exists at least one blocking pair which makes the matching unstable. According to Theorem 1, we know that blocking pairs do not exist in Φ^*, so it is Pareto optimal. ■

V. Dedicated Cache Space Allocation and Management

A. Problem Formulation

After selecting the core nodes in previous section and completing the matching, we then need to find the optimal dedicated cache space allocation at core nodes. First, we need to get the maximum cache revenue that can be brought by the dedicated cache space on core nodes. For a given core node *n* with a m_n size dedicated cache space, its matched edge nodes set is $\Phi(n)$. For each edge node $(\forall v \in \Phi(n))$, we assume that its distance to the content server of paying CP *i* and core node *n* can be expressed as d_{v-cp_i} and d_{vn}, respectively, and the

Algorithm 1: Many-to-One Matching Between Edge Nodes and Core Nodes

```
Initialization:
Initialize the matched pairing list $\Phi(cn, en) = \varnothing$ to record;
Initialize the set of unmatched edge nodes $UM\_EN = EN$ and the set of unmatched core nodes $UM\_CN = CN$;
Set up the preference lists $PF_{CN}, PF_{EN}$;
Set up the received requests of each node $r\_req() = \varnothing$;
Stage 1:
while $UM\_CN \neq \varnothing$ do
    According to $PF_{CN}(m)$, each core node $cn_m \in UM\_CN$ sends request to its current most preferred edge node $en_n \in UM\_EN$;
    for each $en_n \in UM\_EN$ do
        $en_n$ select core node $cn_m \in r\_req(en_n)$ among the received requests, which maximizes $u_{EN,n}(m)$;
        $\Phi = \Phi + (cn_m, en_n)$;
        $UM\_CN = UM\_CN - \{cn_m\}$;
        $UM\_EN = UM\_EN - \{en_n\}$;
    end
end
Stage 2:
repeat:
Based on $PF_{CN}$, each core node send requests to its at most k preferred edge nodes that have never rejected it before;
each edge node $en_n \in EN$ select core node $cn_m \in r\_req(en_n) \cup \Phi(en_n)$ which maximizes $u_{EN,n}(m)$;
if $|\Phi(cn_m)| < k$ then
    if $\Phi(en_n) \neq \varnothing$ then
        $\Phi = \Phi - (en_n, \Phi(en_n))$;
    end
    $\Phi = \Phi + (cn_m, en_n)$;
    edge node $en_n$ reject other core nodes' requests;
end
if $|\Phi(cn_m)| = k$ and $(cn_m, en_n) \notin \Phi$ then
    core node $cn_m$ select edge node $en_q$ maximizing $(u_{CN,m}(n) - u_{CN,m}(q)) > 0, q \in \Phi(cn_m)$;
    if $\Phi(en_n) \neq \varnothing$ then
        $\Phi = \Phi - (en_n, \Phi(en_n))$;
    end
    $\Phi = \Phi + (cn_m, en_n) - (cn_m, en_q)$;
    edge node $en_n$ reject other core nodes' requests;
end
stage 2 ends when no core nodes send new requests;
output the final matching $\Phi^* = \Phi$;
```

local users' requests for contents from the paying CP can be expressed as req_v. We define the cache revenue that the edge node *v* can get from core node *n* as $req_v * (d_{v-cp_i} - d_{vn})$, that is the sum of the length of the transmission path saved by all repeated requests for the contents of paying CP. Then overall cache revenue R_n at core node *n* can be expressed as

$$R_n = \sum_{v \in \Phi(n)} req_v * \left(d_{v-cp_i} - d_{vn}\right) * \int_0^{m_n/s} g(x)\ dx, \tag{8}$$

$$g(x) = \frac{(x+\varepsilon)^{-\alpha}}{\int_0^W (w+\varepsilon)^{-\alpha} dw}, \tag{9}$$

where *s* is the size of a single content and we assume all chunks have the same size here. *g*(*x*) is the probability density function of the distribution of content popularity, *x* represents the popularity level and the probability of users requesting the *X*-th popular content is $\int_{X-1}^{X} g(x)dx$, *W* represents the total number of paying CP's contents. The parameters ε and α represent both the skewness and the centrality of the popularity distribution, which are the two opposite properties of the distribution. Skewness is positively correlated with the value of ε, while the centrality is positively correlated with the value of α. But in most cases, we usually set ε to 0 and modify the distribution by controlling the value of α.

For the paying CP, assume that its total budget is *C*, the unit price of cache space at the candidate node *n* is p_n, we can get the budget constraint as

$$\sum_n m_n * p_n \leq C. \tag{10}$$

Generally, the more budget a CP spend, the larger the cache space and the better the caching service will be obtained. Therefore, we assume that the total cost of paying CP in dedicated cache space allocation can always reach *C*. Thus, our optimization goal is to maximize the sum of all core nodes' cache revenues under the given total budget *C* and the rentable size of cache space, which can be expressed as

$$max\ \ R_{total} = \sum_n R_n \tag{11}$$

$$subject\ to\colon\ 0 \leq m_n \leq M_n, \tag{11.a}$$
$$\sum_n m_n * p_n = C, \tag{11.b}$$
$$0 \leq m_n \leq s * W, \tag{11.c}$$

where R_{total} represents the total cache revenue of all core nodes' the dedicated cache space and M_n represents the maximum size of cache space that the paying CP can rent at core node *n*.

B. Problem Transformation and Solution

Obviously this problem is concave and have both equality and inequality constraints, which is hard to be solved directly. Thus, we consider converting it into a convex problem to solve and we set

$$f(m) = -R_{total}. \tag{12}$$

Then we can add the inequality constraints to the objective function and transform Eq. (12) into an equality constraint problem as following

$$min\ \ f(m) + \sum_n I(f_n(m)) \tag{13}$$

$$subject\ to\colon\ Pm = \sum_n p_n * m_n = C, \tag{13.a}$$

where $f_n(m) = m_n - M_n$, *I* is non-positive and defined as:

$$I(z) = \begin{cases} 0, & if\ z \leq 0 \\ \infty, & otherwise \end{cases} \tag{14}$$

I(*z*) is not differentiable, so we use the approximate function $I'(z)$ instead of *I*(*z*), where

$$I'(z) = -(1/t)log(-z),\ dom(I'(z)) = -\mathbb{R}_{++}. \tag{15}$$

Algorithm 2: Iterative Algorithm for the Approximation Problem

```
Initialization:
Initialize the approximation problem $t * f(m) + \phi(m)$
Initialize the maximum tolerance $\delta > 0$;
Initialize the original value of $m$, $t$ and $e$, $e > 1$;
Iteration: while $m/t < \delta$ do
    Under the constraint $Pm = C$, use Newton method
    to find the minimum point $m^*(t)$ of $t * f(m) + \phi(m)$;
    $m := m^*(t)$;
    $t := e * t$;
end
output the approximate optimal solution.
```

$t > 0$ is a parameter that determines the accuracy of the approximation between problem in Eq. (13) and the original problem in Eq. (11). The bigger the value of t is, the higher the accuracy of the approximation will be. Substituting Eq. (15) into Eq. (13) and multiplying the objective function by t, we can get

$$min \quad t * f(m) + \phi(m) \tag{16}$$

$$subject\ to: \ Pm = C, \tag{16.a}$$

where $\phi(m) = -\sum_n log(-f_n(m))$. Then the optimal solution of Eq. (16) is the approximate optimal solution of the original problem when t is large enough and the specific proof is introduced in the Appendix. The iterative process to solve Eq. (16) is shown in Algorithm 2.

On this basis, when we need to allocate cache space in ICN, the specific execution steps are as follows. First, model the problem as the original optimization problem in Eq. (11) based on the collected network information. Then transform it into the approximation problem in Eq. (13) with only equality constraints. After that, use Algorithm 2 to find the solution of the equivalent problem in Eq. (16) which has the same optimal solution set as the approximation problem in Eq. (13). Then the optimal solution we get from Algorithm 2 is the approximate optimal solution of the original problem, that is, the optimal cache space allocation under the current network conditions is calculated.

VI. System Implementation

Each node in NDN has three basic components, the content store (CS), the pending interest table (PIT) and the forwarding information base (FIB). CS stores copies of different contents. When the node receives an interest packet, it will first retrieve its CS. If there is a corresponding content, it directly returns. If not, the interest packet will be further forwarded to next hop. PIT records the sources of the received interests. When the corresponding content is returned to the node, it will deliver the content to the sources of interest packets using the information recorded in PIT. FIB is similar to the routing table in traditional IP network and it is responsible for querying the next hop of the interest packet forwarding. Therefore, each NDN node has independent storage, routing, and forwarding functions.

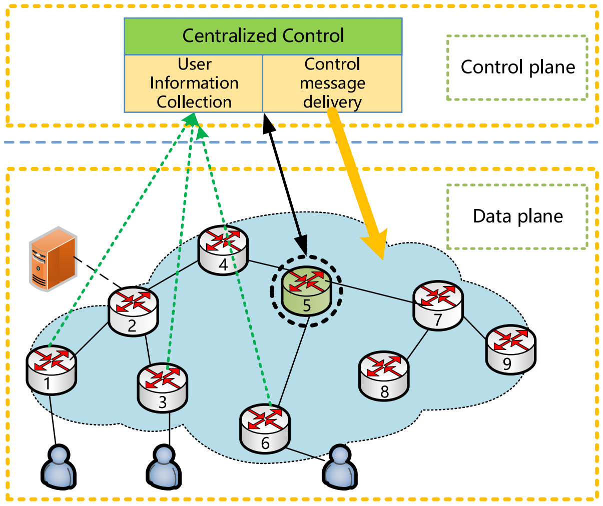


Fig. 2. System implementation.

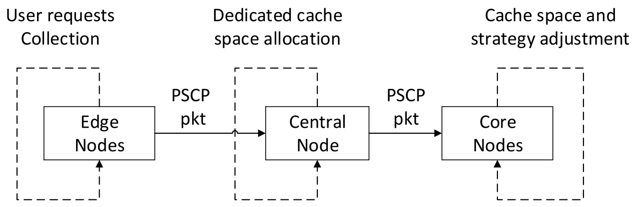


Fig. 3. Control message interaction.

However, these basic functions are not enough to support the implementation of our scheme. As shown in Fig. 2, we use a central-control idea to implement our service prioritization scheme. The entire network is logically divided into two layers: the control plane and the data plane. The control plane is abstractly deployed on a fixed central node like the node 5 in Fig. 2. It is mainly responsible for two tasks, namely the calculation and allocation of dedicated cache space and the issuance of control messages. All router nodes in the network constitute the data plane, including core nodes, edge nodes and other nodes. Among them, edge node is responsible for the collection and periodic reporting of user request information, and core node adjusts its local cache space allocation and caching strategy according to the received control message.

In addition to the basic user request and data acquisition process of ICN, we mainly need to realize the interactive process of control information in our scheme, as it is shown in Fig. 3. Since the control plane and the data plane need to frequently exchange information, we first define a new control packet type named PSCP (Prioritized Service Control Packet) to carry user information and control messages, which is shown in Fig. 4. The type field is a 2-bit field equals to 0 or 1 which respectively mean that the packet carries user information or control message. Then the next two fields are 7-bit fields corresponding to the identity of paying CPs and the packet generator. The timestamp is a 16-bit field which ensures the freshness of the information. The last two fields both have 16-bit length, which are respectively used to carry

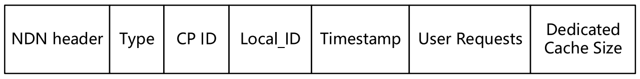


Fig. 4. PSCP packet format.

```
Algorithm 3: Caching Strategy for Core Node n
Input:
1)Content received: $c_i$;
2)Caching strategy at node n: $\mu_n$;
3)Dedicated cache space $S_{DC}$, normal cache space $S_{NC}$;
4)$\mu_n(c_i) = 1$ means content $c_i$ satisfies the cache requirement of node n;
when core node n receives content $c_i$
if $c_i$ is not cached locally then
    if $S_{NC}$ is not full then
        cache $c_i$ in $S_{NC}$;
    else
        if $\mu_n(c_i) = 1$ in $S_{NC}$ then
            cache $c_i$ in $S_{NC}$;
            $S_{NC}$ is full, content $c_j$ is evicted;
            if $c_j$ belongs to the paying CP then
                if $\mu_n(c_j) = 1$ in $S_{DC}$ then
                    caching $c_j$ in $S_{DC}$;
                end
            end
        else
            if $c_i$ belongs to the paying CP then
                if $\mu_n(c_i) = 1$ in $S_{DC}$ then
                    caching $c_i$ in $S_{DC}$;
                end
            end
        end
    end
end
forward $c_i$ to next hop according to the PIT;
```

edge node's user request statistics and the optimal dedicated cache size that the central node allocate to a specific core node.

The detailed interaction between the control plane and the data plane consists of the following three steps.

1) Edge Nodes: First, each edge node records the local users' requests for paying CPs' contents. Then edge node periodically encapsulates the statistical information in the most recent period of time in the PSCP and sends it to the central node where the control plane is deployed. The type field of PSCP is set to be 0 and the total number of requests from local users in the most recent period will be recorded in the corresponding field.

2) Central Node: Periodically, when the central node receives user information from all edge nodes, it calculates the current optimal dedicated cache space allocation based on the information and the matching relationship between core nodes and edge nodes. After that, each calculated cache space size will be carried in the PSCP with the type field set to 1 and sent to the corresponding core node.

3) Core Nodes: When receiving the control message, core node adjusts its local cache space allocation and the corresponding cache strategy. The caching decision process is detailed in Algorithm 3. Specifically, when making a cache decision, the decision is first made in the normal cache space.

TABLE II
EXPERIMENTAL PARAMETERS

Parameter name	Parameter value
Number of network nodes	50
Number of clients	48
Number of content servers	2
Node's cache space size	100-500
User's request rate	100-500 interest/s
Link delay	1-5ms
α parameter	0.5-1.5

For the content that cannot be cached or the content that is evicted, core node evaluates whether the content belongs to the paying CP. And if it is, another cache decision will be made in the dedicated cache space.

VII. PERFORMANCE EVALUATION

A. Simulation Environments

In this section, we evaluate the performance of our proposed scheme. We conduct experiments based on ndnSIM [19] (version 2.7, a ns-3 based NDN simulator) under the Ubuntu 18.04 virtual machine. We use the BRITE [36] topology generator to generate an experimental network topology that consists of 100 nodes based on the Waxman model, among which there are 48 clients, 2 different content servers and 50 network nodes. One of content servers represents the paying CP and another is a normal CP, they both own 10,000 different contents. There are 20,000 different contents in the network, each content has the same size and the popularity of contents is subject to distribution in Eq. (9). The request frequency of each user node is subject to Poisson distribution and the average cache size is same for each node. The specific experimental parameters are shown in Table II.

B. Simulation Results

In order to prove the effectiveness of the scheme, we evaluate it from multiple aspects under different experimental parameter settings. Through the implement of the service prioritization scheme, we aim to reduce the average content acquisition delay while increasing the cache hit ratio, and thus improve the overall QoE of users.

We evaluate the importance of each node in the network based on the topology generated by BRITE [36] as shown in Fig. 5, and select 5 nodes with the top 10% of node importance as core nodes. The 48 client nodes are randomly generated, so there are 17 edge nodes with user access and 28 ordinary nodes in the final network topology. Then we use the matching algorithm designed in Section III to pair the core nodes and edge nodes. In the experiment, we set $k = 10$, that is, a core node is paired with no more than 10 edge nodes. Finally, the number of edge nodes paired by each core node is shown in Table III. In practical applications, we should also take topology and link bandwidth into consideration, and set a suitable upper limit for node pairing to avoid a core node match with too many edge nodes, which may cause potential network congestion.

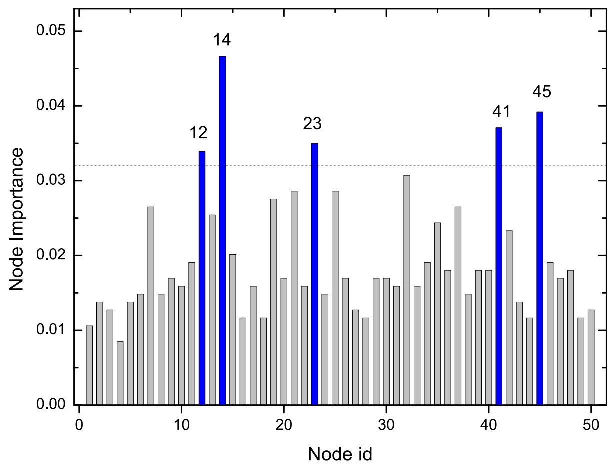


Fig. 5. Node importance.

TABLE III
NUMBER OF CORE NODES' MATCHED NODES

Core node	12	14	23	41	45
Number of matched edge nodes	1	7	4	2	3

Simulation I: In the first part, we evaluate the performance improvement brought by the proposed service prioritization scheme. We test the schemes under two different caching strategies of LEC (Leave copy everywhere) and LFU (Least frequently used) respectively. We select the average content acquisition delay and cache hit ratio as performance indicators. The cache hit ratio refers to the probability that a user's request will be satisfied on a network node instead of a content server. The average content acquisition delay refers to the average time from the user send a content request to receive the corresponding data packet.

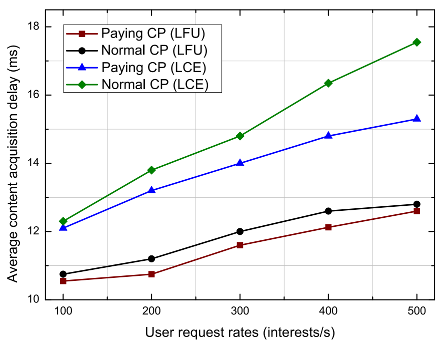
We first fix the cache size of the node to 400 and the content popularity distribution's α parameter to 0.7, and then test the average content acquisition delay and cache hit ratio when users obtain contents from different content providers under different user request rates. As we can see from Fig. 6(a) and Fig. 7(a), through the implement of service prioritization scheme, users can get better experience in terms of both delay and cache hit ratio when requesting the contents of paying CP. And this advantage continues to expand as user request rates increase because our scheme allows paying CP to occupy more in-network cache resources. The increase in network traffic loads will bring more packet loss and congestion, making the delay of end-to-end retransmission unacceptable. On the contrary, the greater the delay of end-to-end transmission is, the higher the benefits that in-network caches can bring. But to be noticed, we can see from Fig. 6 and Fig. 7 that the service prioritization scheme with LCE caching strategy always cannot achieve a well performance. This is caused by the limitations of the LCE strategy itself, which cannot guarantee the effective use of cache resources. So when deploying and designing our scheme, we should also choose the most appropriate caching strategy if permitted.

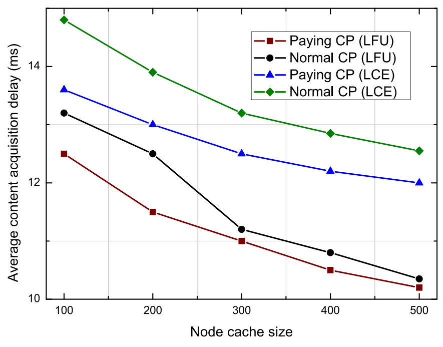
Then we set the user's request rate to 400interests/s and the content popularity distribution's α parameter to 0.7, and compare the cache hit ratio of the contents under different node cache size settings. As illustrated in Fig. 6(b) and Fig. 7(b), paying CP always has a better performance. Because the service prioritization scheme allows the paying CP to occupy more cache space, so it can cache more contents and undoubtedly can obtain a higher cache hit ratio under the same conditions. However, a larger node cache size will reduce the advantages of paying CP. User requests for contents follow the popularity distribution in Eq. (9), and only a small part of the contents is highly popular. When the node cache is large enough, both CPs have enough space to cache popular contents. Compared with the normal CP, the benefits of low-popular content cached by paying CP are very limited. When the node cache size continues to increase, it is not difficult to infer that this gap will get smaller and smaller until it disappears. On the one hand, this conclusion confirms one of the important consideration of our scheme, that is, in a network with limited resources, we can greatly improve the quality of caching service for paying CP through increasing its cache space. On the other hand, this also shows that when deploying dedicated cache space, more is not always better. When a certain threshold is reached, additional cache deployment can only bring very limited benefits while costing more.

After that, we set the user's request rate to 400interests/s and node cache size to 400, then we evaluate the performance of our service prioritization scheme under different user request distributions. As it is shown in Fig. 6(c) and Fig. 7(c), although the service prioritization scheme has always achieved better performance, but this advantage keeps decreasing until it disappears with the parameter α increase. This is because with the increase of α, the diversity of users' requests continues to decrease. Then finally almost all popular contents can be cached on each node, so that all schemes can get the best performance.

Simulation II: After that, we compare our service prioritization scheme with the existing diff-serv scheme [11] which is the most representative one in NDN that jointly consider forwarding and caching. Here we only conduct comparative experiments with its caching model "diff-caching" which is somewhat similar to our scheme. In diff-caching, CP controls the size of the cache window, corresponding to the number of contents that will be cached on a priority basis at network nodes. At the same time, it can be seen that the CP has leased some cache space on these nodes that provide prioritized caching services.

First, we assume that diff-caching rents part of the cache space on all nodes in the network according to its original scheme. We assume that the unit space rental price of non-core nodes is equal to the average value of the prices of core nodes, and give diff-caching the same total budget as our scheme. We fix the cache size of the node to 400 and the content popularity distribution's α parameter to 0.7, set LFU as the default caching strategy on each node, then we evaluate the two schemes under different user request rates. As the results are shown in Fig. 8 and Fig. 9, the paying CP can always get better content acquisition delay and cache hit ratio in our


(a) Delay changes with the user request rate


(b) Delay changes with the node cache size

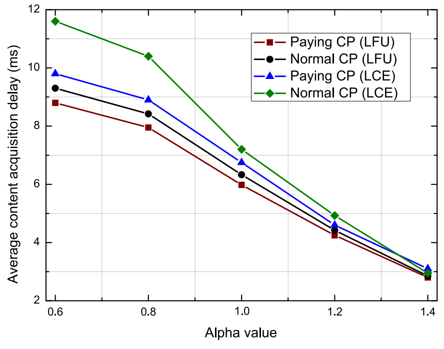
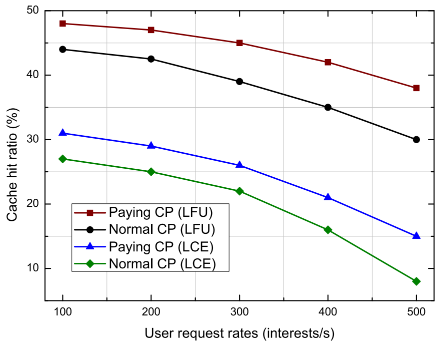
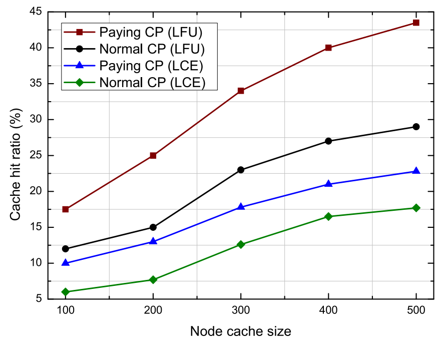

(c) Delay changes with the alpha value

Fig. 6. Average content acquisition delay under different network conditions.


(a) Hit ratio changes with the user request rate


(b) Hit ratio changes with the node cache size

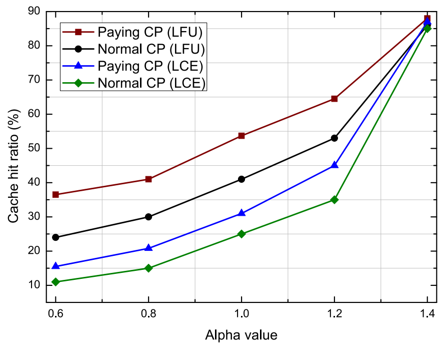

(c) Hit ratio changes with the alpha value

Fig. 7. Cache hit ratio under different network conditions.

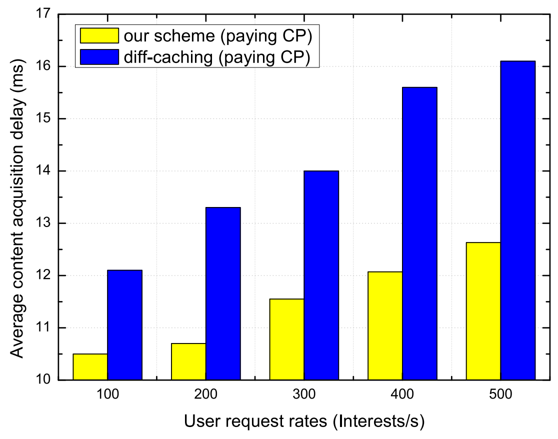


Fig. 8. Delay of different schemes.

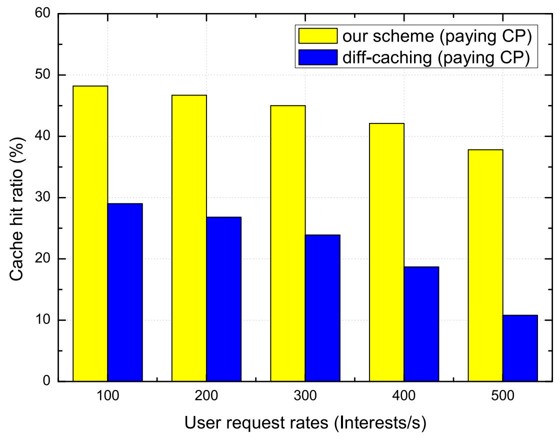


Fig. 9. Cache hit ratio of different schemes.

scheme. This shows the rationality and effectiveness of the core nodes selected in our scheme also confirms why we only deploy cache space on some core nodes. When the budget is limited, deploying small cache space on all nodes can only bring little improvement, and it is far inferior to deploying larger cache space on a few key nodes.

Second, we assume that diff-caching also only rents cache space on core nodes. We set the user's request rate to 400interests/s and node cache size to 400. Then we let the users' requests of the edge nodes move over time, and measure the current average acquisition delay for the paying CP every 2 minutes under the two schemes. As we can see from Fig. 10, our scheme can self adjust according to the network conditions, and can always achieve stable and good performance. Based on the above two experiments, we can conclude that our service prioritization scheme can not only guarantee well performance, but also has better dynamics and adaptability.

VIII. Conclusion

In this paper, we proposed a new service prioritization scheme in ICN to provide different caching services for heterogeneous content providers. The main idea of our scheme

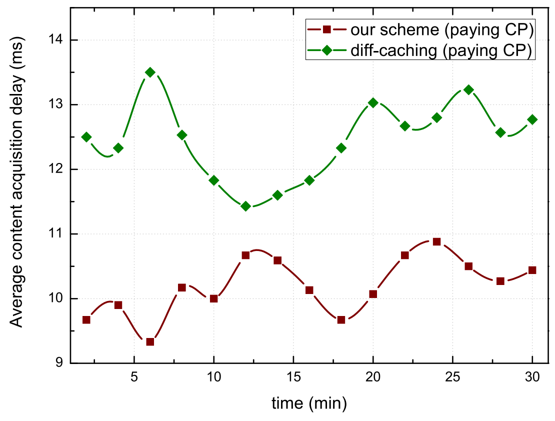


Fig. 10. Delay changes with time.

is to allocate dedicated cache space for paying CPs, and then provide prioritized caching service for their contents. To do so, we first selected a group of potential deployment nodes as core nodes and paired them with all the edge nodes using two sided many-to-one matching, and then calculated the optimal dedicated cache space allocation for each core node based on user information. After that, we proposed a practical implementation of our scheme and designed the prioritized caching strategy for those core nodes. Finally, we evaluated the performance of our service prioritization scheme in ndnSim simulator and the results verified the effectiveness of our scheme.

Appendix

In order to prove that the problem in Eq. (16) has the same optimal solution as the original problem in Eq. (11), we first have

$$\nabla\phi(m) = \sum_n \frac{1}{-f_n(m)} \nabla f_n(m). \tag{17}$$

For $\forall t > 0$, the above optimization problem in Eq. (16) has an unique optimal solution $m^*(t)$, the set $\{m^*(t)\}$ composed of optimal solutions is the central path. And there exists υ' which satisfies

$$\nabla f(m^*(t)) + \nabla\phi(m^*(t)) + P^T\upsilon' = 0. \tag{18}$$

Define $\lambda^*(t) = -\frac{1}{t*f_n(m^*(t))}$ and $\upsilon^*(t) = \upsilon'/t$. When $\lambda = \lambda^*(t)\ and\ \upsilon = \upsilon^*(t)$, the Lagrangian function can be written as

$$L(m, \lambda, \upsilon) = f(m) + \sum_n \lambda_n f_n(m) + \upsilon^T(Pm - C). \tag{19}$$

When $m = m^*(t)$, Eq. (19) takes a minimum value, which means that $\lambda^*(t)$ and $\upsilon^*(t)$ are dual feasible solutions. So the dual function $g(\lambda^*(t), \upsilon^*(t))$is bounded:

$$\begin{aligned} g(\lambda^*(t), \upsilon^*(t)) &= f(m^*(t)) + \sum_n \lambda^*(t) f_n(m^*) \\ &\quad + \upsilon^*(t)^T(Pm^*(t) - C) \\ &= f(m^*(t)) - m/t. \end{aligned} \tag{20}$$

Thus, the gap between the approximate solution $g(\lambda^*(t), \upsilon^*(t))$ and the optimal solution p^* of the original problem is *m*/*t*. When $t \to \infty$, there is $m/t \to 0$, so we can get the asymptotic optimal solution of the original problem. That is, when $t \to \infty$, the solution of problem in Eq. (16) is the same as the solution of the original problem.

Acknowledgment

The authors sincerely thank the Editor, Dr. Azzam Mourad, and the anonymous referees for their valuable suggestions that have led to the present improved version of the original manuscript.

References

[1] "Cisco visual networking index: Forecast and trends, 2017–2022," Cisco, San Jose, CA, USA, White Paper, 2018.
[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
[3] L. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.
[4] T. Koponen *et al.*, "A data-oriented (and beyond) network architecture," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun. (SIGCOMM)*, 2007, pp. 181–192.
[5] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrelld, B. Ahlgrene, and H. Karl, "Network of information (NetInf)—An information-centric networking architecture," *Comput. Commun.*, vol. 36, no. 7, pp. 721–735, 2013.
[6] W. Li, S. Wang, Y. Xu, and S. Lu, "Charging on the route: An online pricing gateway congestion control for ICNs," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 239–250, Mar. 2020.
[7] S. Wang, J. Bi, J. Wu, and A. V. Vasilakos, "CPHR: In-network caching for information-centric networking with partitioning and hash-routing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2742–2755, Oct. 2016.
[8] S. Lee, I. Yeom, and D. Kim, "T-Caching: Enhancing feasibility of in-network caching in ICN," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1486–1498, Jul. 2020.
[9] B. Nour, H. Khelifi, H. Moungla, R. Hussain, and N. Guizani, "A distributed cache placement scheme for large-scale information-centric networking," *IEEE Netw.*, vol. 34, no. 6, pp. 126–132, Nov./Dec. 2020.
[10] R. Huo, R. Xie, H. Zhang, T. Huang, and Y. Liu, "What to cache: Differentiated caching resource allocation and management in information-centric networking," *China Commun.*, vol. 13, no. 12, pp. 261–276, Dec. 2016.
[11] Y. Kim, Y. Kim, J. Bi, and I. Yeom, "Differentiated forwarding and caching in named-data networking," *J. Netw. Comput. Appl.*, vol. 60, pp. 155–169, Jan. 2016.
[12] Y. Kim, Y. Kim, and I. Yeom, "Differentiated services in named-data networking," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, 2014, pp. 452–457.
[13] Y. Bernet *et al.*, "A framework for integrated services operation over diffserv networks," IETF, RFC 2998, 2000. Accessed: Aug. 2021. [Online]. Available: https://www.ietf.org/rfc/rfc2998.txt
[14] T. Aimoto and S. Miyake, "Overview of DiffServ technology: Its mechanism and implementation," *IEICE Trans. Inf. Syst.*, vol. 83, no. 5, pp. 957–964, 2000.
[15] P. Flegkas, P. Trimintzios, and G. Pavlou, "A policy-based quality of service management system for IP DiffServ networks," *IEEE Netw.*, vol. 16, no. 2, pp. 50–56, Mar./Apr. 2002.
[16] L. Han, Y. Qu, L. Dong, and R. Li, "A framework for bandwidth and latency guaranteed service in new IP network," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, 2020, pp. 85–90.
[17] D. Aureli, A. Cianfrani, A. Diamanti, J. M. S. Vilchez, and S. Secci, "Going beyond DiffServ in IP traffic classification," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Budapest, Hungary, 2020, pp. 1–6.
[18] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 95–107, Jan. 2016.

[19] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2: An updated NDN simulator for NS-3," NDN Rep. NDN-0028, 2016. Accessed: Aug. 2021. [Online]. Available: https://named-data.net/wp-content/uploads/2013/07/ndn-0028-1-ndnsim-v2.pdf

[20] M. A. Ridwan, N. A. M. Radzi, W. S. M. W. Ahmad, F. Abdullah, M. Z. Jamaludin, and M. N. Zakaria, "Recent trends in MPLS networks: Technologies, applications and challenges," *IET Commun.*, vol. 14, pp. 177–185, Jan. 2020.

[21] S. Hasija, R. Mijumbi, S. Davy, A. Davy, B. Jennings, and K. Griffin, "Domain federation via MPLS and SDN for dynamic, real-time end-to-end QoS support," in *Proc. 4th IEEE Conf. Netw. Softw. Workshops (NetSoft)*, Montreal, QC, Canada, 2018, pp. 177–181.

[22] K. Benzekki, A. El Fergougui, and A. E. Elalaoui, "Software-defined networking (SDN): A survey," *Security Commun. Netw.*, vol. 9, no. 18, pp. 5803–5833, 2016.

[23] Y. Xu and D. Xu, "Maximizing profit of network InP by cross-priority traffic engineering," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, 2018, pp. 1–6.

[24] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun. (SIGCOMM)*, 2013, pp. 3–14.

[25] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 3, pp. 286–299, Sep. 2013.

[26] Y. Li, H. Xie, Y. Wen, C.-Y. Chow, and Z.-L. Zhang, "How much to coordinate? optimizing in-network caching in content-centric networks," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 3, pp. 420–434, Sep. 2015.

[27] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF based routing protocol for named data networking," NDN Rep. NDN-003, 2012. Accessed: Aug. 2021. [Online]. Available: https://www.named-data.net/techreport/TR003-OSPFN.pdf

[28] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data link state routing protocol," in *Proc. 3rd ACM SIGCOMM Workshop Inf. Centric Netw. (ICN)*, 2013, pp. 15–20.

[29] F. Li, L. Zhang, Y. Liu, and Y. Laili, "QoS-aware service composition in cloud manufacturing: A Gale–Shapley algorithm-based approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 7, pp. 2386–2397, Jul. 2020.

[30] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[31] K. You, R. Tempo, and L. Qiu, "Distributed algorithms for computation of centrality measures in complex networks," *IEEE Trans. Autom. Control*, vol. 62, no. 5, pp. 2080–2094, May 2017.

[32] A. Mauleon, V. J. Vannetelbosch, and W. Vergote, "Von Neumann–Morgenstern farsightedly stable sets in two-sided matching," *Theor. Econ.*, vol. 6, no. 3, pp. 499–521, 2011.

[33] Y. Liu, X. Li, F. R. Yu, H. Ji, H. Zhang, and V. C. M. Leung, "Grouping and cooperating among access points in user-centric ultra-dense networks with non-orthogonal multiple access," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 10, pp. 2295–2311, Oct. 2017.

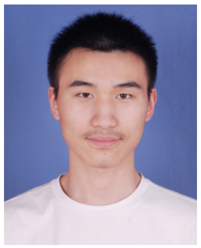
[34] J. Zhao, Y. Liu, K. K. Chai, Y. Chen, and M. Elkashlan, "Many-to-many matching with externalities for device-to-device communications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 1, pp. 138–141, Feb. 2017.

[35] M. I. Kamel, W. Hamouda, and A. M. Youssef, "Multiple association in ultra-dense networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[36] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in *Proc. 9th Int. Symp. Model. Anal. Simulat. Comput. Telecommun. Syst. (MASCOTS)*, Cincinnati, OH, USA, 2001, pp. 346–353.

Jin Qin (Graduate Student Member, IEEE) received the B.S. degree from the Department of Automation, University of Science and Technology of China, Hefei, China, in 2015, where he is currently pursuing the Ph.D. degree in information security with the School of Cyber Security. His research interests include next-generation Internet architecture design and performance optimization.

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC) in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science, USTC, in 2007. From May 2012 to May 2013, he was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida. He is currently a Professor with the School of Cyber Security, USTC. His research interests include next-generation Internet architecture design, transmission optimization and network security. He serves on the Editorial Board of several journals, including the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, *IEEE Communications Magazine*, and IEEE NETWORK. He is an IET Fellow.



Jian Li (Member, IEEE) received the B.S. degree from the Department of Electronics and Information Engineering, Anhui University in 2015, and the Ph.D. degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC) in 2020. From November 2019 to November 2020, he was a Visiting Scholar with the Department of Electronic and Computer Engineering, University of Florida. He is currently a Postdoctoral Researcher with the School of Cyber Security, USTC. His research interests include wireless communications, satellite networks, and next-generation Internet.

Qibin Sun (Fellow, IEEE) received the Ph.D. degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, in 1997, where he is currently a Professor with the School of Cyber Security. He has published more than 120 papers in international journals and conferences. His research interests include multimedia security, network intelligence and security and so on.

Jun Lu received the bachelor's degree from Southeast University in 1985, and the master's degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, in 1988, where he is currently a Professor. His research interests include theoretical research and system development in the field of integrated electronic information systems. He is an Academician of the Chinese Academy of Engineering.