IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 15, 2020, accepted October 22, 2020, date of publication October 28, 2020, date of current version November 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034329

A Cache Placement Strategy Based on Compound Popularity in Named Data Networking

YIQI GUI AND YONGKANG CHEN
College of Information Engineering, Yangzhou University, Yangzhou 225009, China
Corresponding author: Yiqi Gui (yqgui@yzu.edu.cn)

This work was supported in part by the Natural Science Foundation of Jiangsu Province under Grant bk20150459.

ABSTRACT The built-in pervasive caching is one of the most important features of named data networking (NDN), which can provide effective data delivery even in presence of short-lived and intermittent connectivity. NDN-caching can reduce the expected flood of global data traffic by providing cache storage at intermediate nodes for transmitted contents, making data broadcasting in an efficient way. It also reduces the content delivery time by caching popular content close to consumers. In this article, a cache placement strategy is proposed based on compound popularity (content popularity and node popularity). The proposed scheme aims to enhance the reuse rate of data packets by jointly considering the content popularity and the node popularity in a period of time. Meanwhile, content popularity can be obtained by a lightweight method in our scheme. Performance evaluation in Icarus simulator shows that the proposed scheme performs better in terms of the cache hit ratio, start latency, and link load than some existing strategies.

INDEX TERMS Information centric networking (ICN), named data networking (NDN), content popularity, on-path caching, in-network caching.

I. INTRODUCTION

The Internet plays an increasingly important role in our life with the development of society [1], while internet data traffic has grown explosively in the past few years. According to Cisco VNI forecasts, global IP video traffic will account for 82% of all IP traffic (both commercial and consumer) in 2022, up from 75% in 2017 [2]. The traditional TCP/IP system has many disadvantages in packet distribution, such as poor scalability, low security, and insufficient flexibility [3]. At present, the problem of data packet distribution and sharing through CDN (Content Delivery Network, CDN) [4] and P2P (Peer-to-Peer, P2P) [5] has been alleviated to some extent. The transmission mode of CDN and P2P is still based on the TCP/IP system essentially, which cannot overcome the deficiencies of TCP/IP. To solve the problem, researchers have proposed information centric networking (ICN) network architecture to improve the quality of network services in recent years.

ICN focuses on ''information'' and no longer cares about the specific location of information. In-network caching in ICN can effectively reduce traffic redundancy and source server load. Consumers, Internet Service Provider (ISP), and providers can all obtain huge benefits from in-network caching [6]. Furthermore, In-network caching can also be combined with edge cloud computing [7], Fog Computing [8], Vehicular Ad Hoc Networks [9], Internet of Things (IoT) [10], fifth generation (5G) [11] mobile cellular networks and so on to implement flexible, efficient, and scalable network services.

There are many types of architectures in ICN. Named data networking (NDN) is one of the famous ICN architectures [12]. An effective caching strategy is key to improve the performance of NDN. According to whether the delivery path of the consumer is related to the cache location of the data packet [13], existing caching strategies on NDN can be divided into the on-path caching and off-path caching. On-path caching strategies focus on caching certain content at the appropriate nodes along the delivery path. LCE (Leave Copy Everywhere, LCE) [14], LCD (Leave Copy Down, LCD) [15], and ProbCache (Probabilistic caching, ProbCache) [16] are on-path caching strategies commonly used in NDN. These caching strategies can improve system performance to a certain extent. In off-path caching, data packets are not confined to the delivery path, which can spread beyond the path. Thar *et al.* [17] used consistent hashing as the foundation of a caching strategy. Mun and Lim [18] proposed a cooperative caching strategy based on the Bloom filter.

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

These off-path caching strategies rely on the cooperation of nodes which lead to higher cache utilization than on-path caching in most cases. In particular, off-path caching can perform better than on-path caching strategies in some special application scenarios. For example, off-path caching can better support mobile networks with high-speed mobility requirements [19] in Vehicular Ad Hoc Networks.

Although frequent the cooperation of nodes improves the performance of the off-path caching, it brings a lot of communication overhead which limits the scalability of the network. In this article, we focus on the design of the on-path caching strategy to reduce the communication overhead between nodes. To further improve cache utilization and reduce latency, we propose a lightweight on-path caching strategy. The main contributions of this article are summarized as follows:

1) According to the distribution of consumers in different regions, we analyze the content popularity model and divide it into two parts: global popularity and local popularity. In addition to content popularity, we also consider node popularity, that is, the preference of each content to different nodes on the path during transmission.

2) We combine NDN with edge computing and use a lightweight method to calculate the popular content reasonably. Furthermore, content popularity and node popularity are used efficiently to select appropriate caching nodes for different types of content to maximize the reuse rate of data packets.

3) We consider the impact of different types of network topologies on caching strategies. By using Tiscali-3257 (pan-Europe-commercial ISP) and 7-level complete binary tree topology for simulation experiments, we evaluate and analyze the reasons which lead to the difference in the performance of caching strategies in detail. In the two network topologies, the proposed scheme achieves better performance in the three evaluation metrics: cache hit rate, latency, and link load.

The rest of the paper is organized as follows: Related work is discussed in Section II. Section III describes a system model. Section IV describes the proposed scheme in detail, including the classification of popularity and cache placement. The experimental results are presented in Section V. Section VI concludes this article and provides an outlook for future work.

II. RELATED WORK

To improve the benefits brought by in-network caching, researchers have conducted extensive research on off-path and on-path caching strategies in NDN.

The combination of on-path caching and NDN forwarding mechanism follows the original design principles. On-path caching has attracted extensive attention in the NDN research community and aroused researchers' interest in topics such as content locality [20], reducing cache redundancy [21], and evaluating cache priority by popularity [22], [23]. LCE is the default caching strategy which is also a typical on-path caching strategy. LCE is simple to operate and caches content at all on-path router nodes, while it will cause great cache redundancy, low content diversity, and waste of cache resources in the network. Psaras *et al.* proposed ProbCache, which caches content at on-path nodes with a weighted probability, and the probability is proportional to the number of hops from the caching node to the provider. ProbCache can cache content to edge nodes at a faster speed, reducing cache redundancy to a certain extent. However, ProbCache does not consider content popularity. Laoutaris *et al.* proposed LCD. When a cache hit happens, LCD caches the content objects at the downstream neighbor node of the hit node along the path. LCD can push particularly popular content objects to nodes closer to the consumers gradually. Although LCD improves the cache hit rate to a certain extent. The cache redundancy in the router nodes is not greatly reduced. In particular, LCD performance is very poor in a short period. Chai *et al.* [24] proposed CL4M (betweenness centrality caching, CL4M), which caches content objects at the router node with the greatest betweenness centrality on the path and only caches the same content object once. CL4M can greatly alleviate the problem of high cache redundancy and improve cache performance. However, frequent cache replacement at the nodes with high betweenness centrality will cause serious nodes load. Ren *et al.* [25] designed a distributed caching strategy based on router node cache revenue. This scheme can better improve the caching performance, but the content popularity of each node needs to be calculated separately. Therefore, when the cache strategy is executed in a complex network, the complexity of the algorithm is high [26]. Nguyen *et al.* [27] designed the Progressive Popularity Awareness Cache Scheme (PPCS), which improves the performance of VOD in a tree topology. PPCS also design an autonomous replacement strategy to replace LRU to optimize cache utilization.

In off-path caching, researchers aim to improve the system performance by the cooperation between nodes. Thus, most off-path caching strategies all have different levels of communication overhead. Sourlas *et al.* [28] proposed a caching strategy based on hash routing. This scheme is divided domains into small clusters, and the content in each cluster is discovered by using hash routing. Ullah *et al.* [29] designed a caching strategy for scalable video streaming. This caching strategy layered the tree topology and cached the basic layer of scalable video near the consumer. The higher layers of a scalable video are cached within a specific Round Trip Time range. Zhang *et al.* [30] designed a novel hierarchical proactive caching approach, which considers both vehicular consumer mobility and consumer' future demands. This scheme also predicts consumer's preferences by using the nonnegative matrix factorization technique.

In general, both on-path caching strategy and off-path caching strategy can improve system performance in different application scenarios. Most researchers design an appropriate caching strategy based on actual demand. In this article, one of the key points is to reduce the communication overhead between nodes on the delivery path. Besides, we further

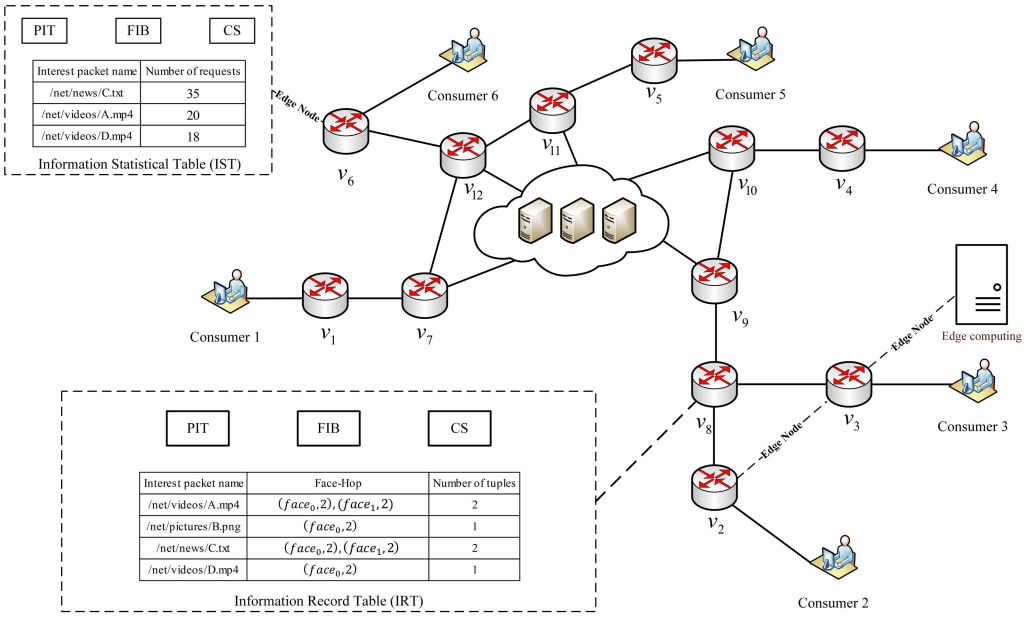


FIGURE 1. System model.

improve the performance of some commonly used on-path caching strategies in NDN to maximize cache benefits and reduce latency by designing a lightweight method.

III. SYSTEM MODEL

In this section, we introduce the NDN communication mechanism in subsection A briefly, and the system model is described in detail in section B.

A. NDN BASIC

The communication of NDN mainly relies on two types of packets: interest packets and data packets. Each router node contains three types of data structures: Pending Interest Table (PIT), Forwarding Information Base (FIB), and Content Store (CS). After receiving an interest packet, the router node will look for the matching data packet in the CS. If the lookup fails, it will check the same pending request in the PIT. Otherwise, the PIT records the interface that received the interest packet and forwards the interest packet by looking up the name in the FIB. The data packet is returned to the consumer by the PIT entry. Meanwhile, the data packet will be cached according to the storage space of the on-path nodes.

B. SYSTEM MODEL

The system model is shown in Fig. 1. The symbols and important abbreviations used in this article are listed in Table 1. $V = \{v_1, v_2, \cdots, v_i\}$ is denoted as the set of all router nodes and each node $v_i \in V$ has a unique id i. E is denoted as the set of edge nodes such as $E = \{v_i | i \in [1, 6]\}$ in Fig. 1. Unlike the traditional NDN model, nodes with edge computing capabilities (like v_3 in Fig. 1) replace the original edge nodes in NDN to calculate the real-time requests of consumers.

TABLE 1. Symbols used to describe the model.

Symbols	Description
V	The set of all router nodes
E	The set of edge nodes
$p(j)$	The request probability of jth popular content
n	The number of current cycles
r_n^C	The number of global requests for content C in the nth cycle
R_n^C	The number of global requests for content C in n cycles calculated by the EWMA model
β	The weight parameter
T_n	The global popularity threshold for the nth cycle
$P_{v_i}^C$	The popularity of node v_i for content C
$B_{v_i}^C$	The cache benefit of content C at node v_i

In particular, each edge node owns the position information of other edge nodes to obtain the global request information by exchanging information with each other regularly. To reduce the overhead, all edge nodes only exchange information at the end of each cycle. Furthermore, each edge node (like v_6 in Fig. 1) is added with an Interest Statistical Table (IST) to record the name and number of interest packets received. At the end of each cycle, the data in the IST of each edge node will be sent to other edge nodes in turn. After sending the data, the data in IST will be cleared.

Information Record Table (IRT) is added at all the router nodes (like v_8 in Fig. 1) to record some basic information about the interest packets that are about to enter the router node. The detailed introduction of IRT will be given

in Section IV. Each node with the same cache capacity can cache the delivered content, and the providers own all the content. Each content block C is a basic cache unit with the same size. The number of current cycles is denoted as n. The number of global requests for content C in the nth cycle is denoted as r_n^C. To minimize the request response time, consumers' request paths are the shortest paths which are calculated by Dijkstra's algorithm. The shortest paths are used to construct the FIB of each router node in NDN. Especially, the construction of FIB is also one of the major challenges of NDN [31], which will not be discussed in detail in this article.

IV. CACHE PLACEMENT STRATEGY BASED ON COMPOUND POPULARITY

The main objectives of our scheme are: (*i*) according to the spatial difference of the requested contents, divide content popularity into global popularity and local popularity for discussion; and (*ii*) finding suitable storage locations for different types of content by combining node popularity. The content popularity is modeled through Zipf [32] function:

$$p(j) = \frac{\theta}{j^{\alpha}} \tag{1}$$

$$\text{s.}t. \quad \theta = \left(\sum_{k=1}^{N} \frac{1}{k^{\alpha}}\right)^{-1} \tag{2}$$

where $p(j)$ is the probability of jth popular content being accessed, and α is the variation factor of content popularity. N is the total number of content categories.

A. GLOBAL POPULARITY

Global popularity is the content popularity of contents requested by all consumers. These request information are collected from multiple cycles. The global popularity in the current cycle is used as a reference standard for content popularity in the next cycle. To reduce the error of calculation results caused by the influence of historical cycle data on content popularity. We adopt two methods: (*i*) the time of a cycle for collecting request information of consumers is set relatively short; (*ii*) EWMA (Exponential Weighted Moving Average) Model [33] is adopted to calculate content popularity. In the EMWA model, the weight of each value decreases exponentially with time, i.e. newer the data item is, the weight becomes heavier. Let R_n^C denotes the global popularity of content C for the next cycle. The calculation formula of R_n^C is as follows:

$$R_n^C = \frac{r_n^C + (1-\beta) r_{n-1}^C + \cdots + (1-\beta)^n r_1^C}{1 + (1-\beta) + \cdots + (1-\beta)^n} \tag{3}$$

where n is the number of current cycles and r_n^C is the number of requests for content C in the current cycle. β [34] is the weight parameter, $\beta \in (0, 1)$. The calculation formula of β is as follows:

$$\beta = \frac{2}{1+n} \tag{4}$$

The global popularity of all content is divided into two types: global popular content (GPC) and global potential popular content (GPPC). GPPC indicates that it may become GPC in the future. When R_n^C is greater than the dynamic popularity threshold T_n, the content C is considered as GPC. The threshold T_n is the average of the number of requests for each content. The calculation of popularity threshold T_n in the nth cycle is:

$$T_n = \frac{\sum_{C=C_1}^{C_M} R_n^C}{M} \tag{5}$$

where $M$is the total number of the content requested categories in n cycles.

The pseudo-code for calculating the GPC and GPPC is shown in Algorithm 1. The global popularity and popularity threshold T_n are calculated by using formula (3) and (5), respectively (lines 1-3). After obtaining GPC (lines 4-6), the remaining contents are sorted in descending order by the number of requests for each content and select the top 25% of the content as GPPC (lines 11–12). At the end of each cycle, algorithm 1 only needs to sort all the data twice to get GPC and GPPC, so the complexity is in the allowable range.

Algorithm 1 Content Popularity Calculating Algorithm in Global Popularity

Input: The all the data in n cycles
1 Use formula (3) to calculate the global popularity of each content;
2 Add the calculation result to the *temp*_1 list;
3 Use formula (5) to calculate T_n;
4 **for** content C in the *temp*_1 list **do**
5 **if** $R_n^C > T_n$ **then**
6 Add content C to the GPC list;
7 **else**
8 Add content C and R_n^C to the *temp*_2 list;
9 **end if**
10 **end for**
11 Sort the *temp*_2 list in descending order;
12 Take the top 25% of the content in the temp_2 list into the GPPC list;
Output: GPC list and GPPC list

B. LOCAL POPULARITY

The purpose of considering the local popularity is to satisfy the special preferences of the local region consumers in the current cycle. For example, content C does not belong to GPC or GPPC in the previous cycles. However, in the current cycle, content C has been requested many times by consumers in a certain local region. If content C is not considered to be cached, subsequent requests of consumers for content C will not be satisfied in time, which will cause a decline in cache performance.

The number of local requests in each cycle is relatively small in comparison to the number of global requests.

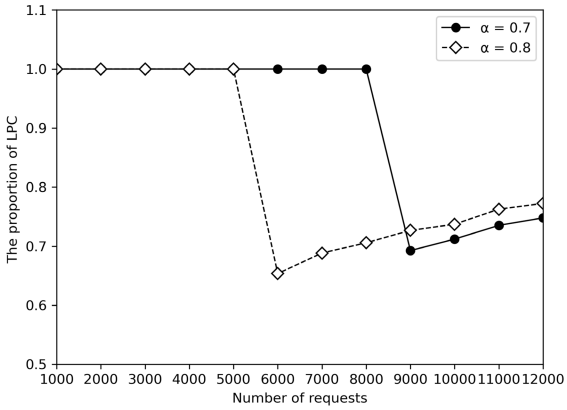


FIGURE 2. Relationship between the proportion of LPC and the number of requests.

Assuming that the local popular content (LPC) is calculated in the same way as GPC and the total number of content objects (content catalog size) is 10,000, Fig. 2 shows the relationship between the proportion of LPC and the number of requests. In the case of the same number of requests, different variation factor α also affect the proportion of LPC. In fact, when the number of requests is small, it is unreasonable that the proportion of LPC is 100%. As the number of requests increases, the proportion of LPC will be less than 100% and stabilize gradually. To avoid this type of error (i.e. the proportion of LPC is 100%), for each local region with a small number of requests in each cycle, we sort the content in descending order and use the first 25% of the content after removing the GPC or GPPC as the LPC.

C. NODE POPULARITY

In a period of time, assuming that content C is not cached in any node on the delivery path. Each node on the delivery path has the same popularity for content C, even if it is a node with large betweenness centrality. If no consumers on other paths have requested content C in this period of time, caching content C in nodes with large betweenness centrality will not improve the reuse rate of data packets. Therefore, we consider the popularity of nodes on the path over a period of time.

Among all the downstream paths of node v_i, we define the number of paths which request content C as the popularity of node v_i for content C, denote $P_{v_i}^C$. The request information of the content C on each downstream path of v_i in a period of time can be obtained from the PIT in each router node. However, the information of the interest packet recorded in the PIT will be deleted when the matching data packet is returned. Consequently, we add an IRT at each node to record basic information of interest packets.

The structure of the IRT is shown in Fig. 1. The tuple (*face*, *hop*) in Face-Hop denotes basic information of an interest packet entering a router node. The *face* is the name of the face which the interest packet enters the router node, and the *hop* is the hop count of the interest packet. The value of $P_{v_i}^C$ is the number of tuples of interest C in the IRT of node v_i. If $P_{v_i}^C$ is greater than one, node v_i is considered as a popular node. As shown in Fig. 1, the popularity of v_8 for */net/news/C.txt* is 2 (i.e. $P_{v_8}^C = 2$).

Especially, if a certain interest packet enters a router node from the same face multiple times, the latest information of this interest packet entering this router node will replace the old information, i.e. there are only tuples with different faces in the Face-Hop of each interest packet. After the end of each cycle, the information in the IRT will be cleared to ensure the timeliness of node popularity.

D. CACHE PLACEMENT

In our scheme, different types of popular content will be cached by different strategies. The LPC will be cached at the edge node of its region to satisfy the needs of local consumers quickly. Combining global popularity and node popularity, we define a cache benefit formula to calculate the cache location of GPC and GPPC. The calculation formula of cache benefit is as follows:

$$B_{v_i}^C = \frac{(1 - x^C)(P_{v_i}^C - 1)}{D_{v_i}^C} \tag{6}$$

where $B_{v_i}^C$ denotes the cache benefit of content C at node v_i. x^C is a binary indicator, $x^C = 0$ means that content C belongs to GPC or GPPC, and $x^C = 1$ implies that content C belongs to LPC. $D_{v_i}^C$ is the sum of the distances between v_i and the consumer who last requested content C on each downstream path of v_i. The smaller $D_{v_i}^C$ means that the closer v_i is to the consumers who request content C, and the content C cached in v_i has a greater cache benefit. The calculation formula of $D_{v_i}^C$ is as follows:

$$D_{v_i}^C = \sum hop_{face_i} \tag{7}$$

where hop_{face_i} is the value of *hop* in each tuple (*face*, *hop*). For example, the value of $D_{v_8}^C$ (i.e. content C refers to */net/news/C.txt*) is 4 in Fig.1.

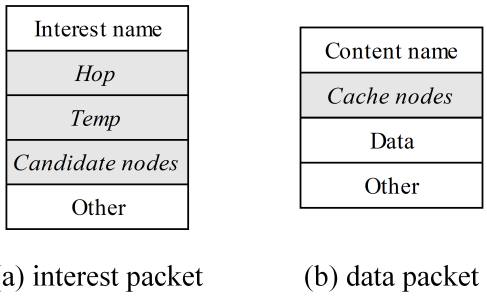


FIGURE 3. Extended structure of interest packet and data packet.

To make the scheme to be successfully implemented in complex networks, we modify the data structure of the original interest packet and data packet. In Fig. 3 (a), three fields are added to the interest packet: *Hop*, *Temp*, and *Candidate nodes*. The *Hop* field records the hop count of the interest packet. The initial value of the *Hop* is 0.

The *Temp* field is a temporary value and its initial value is 0. The *Candidate nodes* field is a null list, which is used to record the ID of the candidate caching nodes. In Fig. 3 (b), only the *Cache nodes* field is added to the data packet.

The pseudo-code for calculating the cache location is shown in Algorithm 2. The IRT of v_i will be updated when an interest packet arrives (line 1). The algorithm 2 aims to determine whether v_i is the node with the largest cache benefit for different types of content and add the ID of the node with the maximum cache benefit to the *Candidate nodes* field of interest packet (lines 2-14).

Algorithm 2 Cache Location Calculation Algorithm Based on Compound Popularity

Input: Interest C and v_i
1 Update IRT of v_i;
2 **if** interest C belongs to LPC **and** v_i is edge node **then**
3 $x^C = 0$;
4 Add v_i to the *Candidate nodes* list of interest C;
5 **else if** interest C belongs to GPC or GPPC **then**
6 $x^C = 1$;
7 **end if**
8 Use formula (6) to calculate $B_{v_i}^C$;
9 **if** $B_{v_i}^C == Temp$ **and** $Temp! = 0$ **then**
10 Add v_i to the *Candidate nodes* list of interest C;
11 **else if** $B_{v_i}^C > Temp$ **then**
12 $Temp = B_{v_i}^C$;
13 *Candidate nodes* list of interest C $\leftarrow$ *null*;
14 Add v_i to the *Candidate nodes* list of interest C;
15 **end if**
Output: The *Candidate nodes* list of interest C

E. CONTENT CACHING STRATEGY

Once a cache hit occurs, the *Candidate nodes* list of the interest packet will be copied to the *Cache nodes* field of the data packet. Therefore, when a data packet returns, it can find the cache location according to the ID of the nodes in its *Cache nodes* list. Note that if the number of nodes in the *Cache nodes* list is more than one, the content will be cached according to the popular type (GPC or GPPC). Case 1: If content C belongs to GPC, it will be cached on all caching nodes. Case 2: If content C belongs to GPPC, it will be cached only at the first caching node of the return path. In addition, if the content C belongs to GPC, and the number of nodes in the *Cache nodes* list is 0. The content C will be cached at the downstream neighbor node of the hit node along the delivery path to ensure that the GPC cache is close to the consumers.

The pseudo-code for content caching strategy is shown in Algorithm 3.

To illustrate the basic content caching mechanism in our scheme, Fig. 4 shows a simplified scenario, which includes three consumers, five router nodes, and one provider. v_1 and v_2 are edge nodes, which have edge computing

Algorithm 3 Content Caching Strategy

Input: Interest C and cache hit node H_{hit}
1 *Cache nodes* $\leftarrow$ *Candidate nodes* list of Interest C;
2 **if** length.*Cache nodes* list $= 1$ **then**
3 **for** v in *Cache nodes* list **do**;
4 Cache content C at v;
5 **end for**
6 **else if** length. *Cache nodes* list > 1 **and** the content C belongs to GPC **then**
7 **for** v in *Cache nodes* list **do**;
8 Cache content C at v;
9 **end for**
10 **else if** length. *Cache nodes* list > 1 **and** the content C belongs to GPPC **then**
11 $v \leftarrow$ *Cache nodes* list.pop;
/* returns the tail element in the list */
12 Cache content C at v;
13 **else if** length. *Cache nodes* list $= 0$ **and** the content C belongs to GPC **then**
14 $v \leftarrow$ the downstream neighbor node of H_{hit} along the delivery path;
15 Cache content C at v;
16 **end if**

capabilities. In the beginning, all router nodes with the same cache capacity do not cache content, and the link delay between router nodes is the same. Assuming that the content C1 belongs to LPC and the content C2 belongs to GPC. At time t1, Consumer 1 and Consumer 2 request content C2 together. At the same time, Consumer 3 requests content C1. Since the content C1 belongs to the LPC, the content C1 will be cached in v_2 when it returns from the provider to Consumer 3. When the interest C2 sent by the Consumer 1 reaches the provider, the interest C2 sent by the Consumer 2 just reaches v_4. According to the NDN forwarding mechanism, the interest C2 sent by the Consumer 2 will be added to the PIT of v_4 and will not continue to be forwarded. According to algorithm 2, the IRT of v_3 and v_4 will be updated. At the end of time t1, content C2 will be cached at v_5 according to algorithm 3. At time t2, Consumer 3 requests content C2. When the interest C2 reaches v_3, the ID of v_3 is added to the *Candidate nodes* list of the interest C2 (i.e. $B_{v_3}^{C2} = 0.25$, $B_{v_3}^{C2} > Temp$) according to algorithm 2. The value of *Temp* changes to 0.25. Subsequently, the interest C2 is forwarded to v_4. Similarly, according to algorithm 2, the ID of v_4 is also added to the *Candidate nodes* list of the interest C2 (i.e. $B_{v_4}^{C2} = 0.25$, $B_{v_4}^{C2} = Temp$ and $Temp! = 0$). Therefore, the content C2 will be cached in v_3 and v_4 according to algorithm 3 when it returns from the v_5 to Consumer 3. The IRT of v_3 and v_4 at the end of time t1 and t2 are shown in the dashed box of Fig. 3. If the three Consumers (Consumer 1, Consumer 2, and Consumer 3) continue to request content C2 later, the content C2 cached in v_3 and v_4 will satisfy their request quickly.

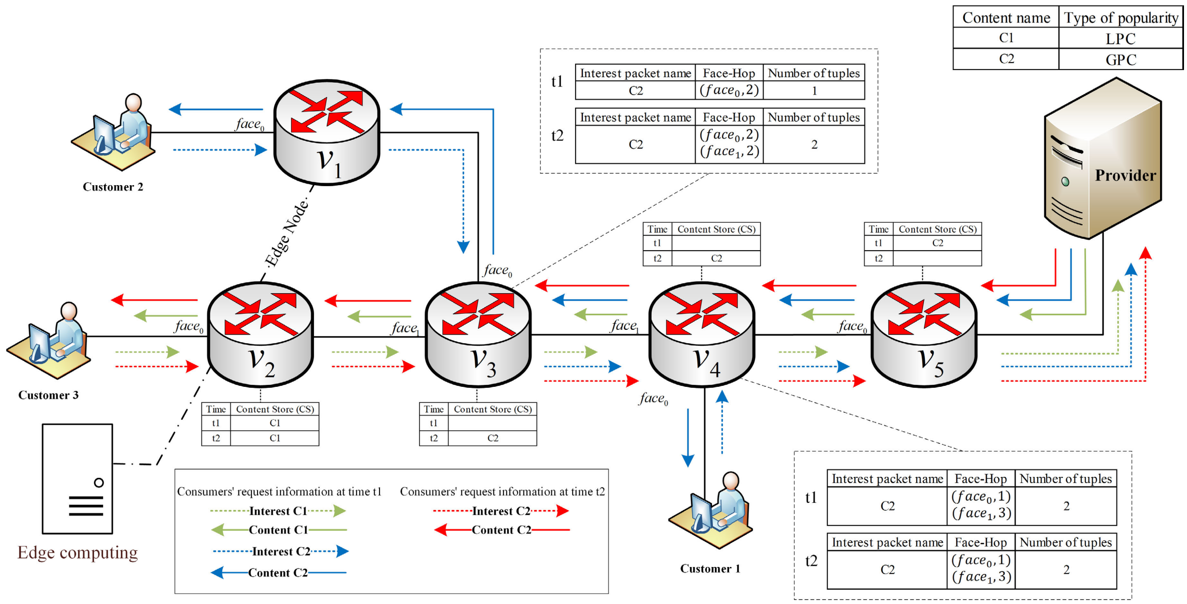


FIGURE 4. Use-case for caching strategy.

V. PERFORMANCE EVALUATION

To evaluate the performance of our scheme, we performed a simulation campaign in two different network topologies by using Icarus simulator [35]: Tiscali-3257 and 7-level complete binary tree topology. The Tiscali-3257 topology has 240 nodes and 404 edges, including 44 provider nodes, 36 consumer nodes, and 160 router nodes. The tree topology has 127 nodes, including the root node as a provider node, 64 leaf nodes as user nodes, and the remaining 62 nodes as router nodes. Only the router nodes on all request paths have cache capabilities. In this article, the 7-level complete binary tree topology is simply called tree topology. The simulation experiments using two network topologies do not mean to the advantages and disadvantages of the network topologies, but to further evaluate and analyze the performance of different caching strategies in different network topologies.

All caching strategies in the simulation experiment use the least replacement policy (LRU) to replace the old cache content. The LRU is considered the most efficient content replacement strategy because of its high performance. All router nodes have the same cache capacity, and the cache to content objects population ratio is S (i.e. the cumulative size of network caches as a fraction of the total content population). The link delay between the router node and the provider node is 34ms, and the link delay between the remaining router nodes is 2ms [36], [37]. The total number of content objects (content catalog size) is 100,000 in the network. The arrival process of consumers follows the Poisson distribution. The content popularity is modeled Zipf distribution with the variation factor of content popularity α. The default value of the Zipf parameter α is 0.8, and it is varied between 0.7 and 1.1. In the tree topology, the only provider provides all content objects, while in Tiscali-3257, all content objects are evenly distributed to all providers. To reduce experimental errors, the cache warms up 50,000 requests and subsequent 250,000 requests are used for performance evaluation. In our scheme, edge nodes obtain global data by exchanging information with each other at the end of each cycle. The number of cycles divisions will directly affect the communication overhead of edge nodes. Figs. 5 (a), 5 (b), and 5 (c) show the three evaluation metrics (i.e. the cache hit ratio, latency, and link load) performance in Tiscali-3257 topology and tree topology against a varying number of cycles for a fixed value of skewness parameter $\alpha = 0.8$ and $S = 0.15$, respectively. The three evaluation metrics performance has a slight improvement in the two network topologies with the increase of the number of cycles. The number of cycles has only a slight impact on the performance of our scheme. The number of cycles for recording request information is equal to 30 in our simulation (i.e. the number of global requests in each cycle is 100,00). Moreover, the proposed scheme can also show good performance under the condition of a small amount of communication overhead. According to the busyness of the system, the proposed scheme can control the communication overhead between edge nodes by adjusting the time of each cycle to exert better performance.

The simulation experiment was carried out 5 times in total, and the average of the five simulation results was taken as the final result. The experimental parameters are shown in Table 2.

A. CACHE HIT RATIO

The cache hit ratio is one of the key metrics to evaluate the performance of NDN. It refers to the response of the content cached in the network within a specific time to the consumer's request. If cache redundancy is low, there will

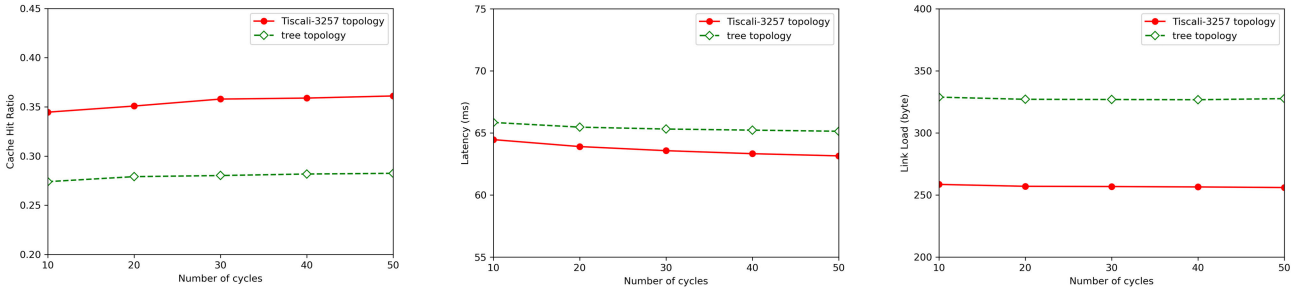


FIGURE 5. Evaluation metrics performance with different number of cycles in Tiscali-3257 topology and tree topology (S = 0.15, $\alpha = 0.8$). (a) Cache hit ratio. (b) Latency. (c) Link load.

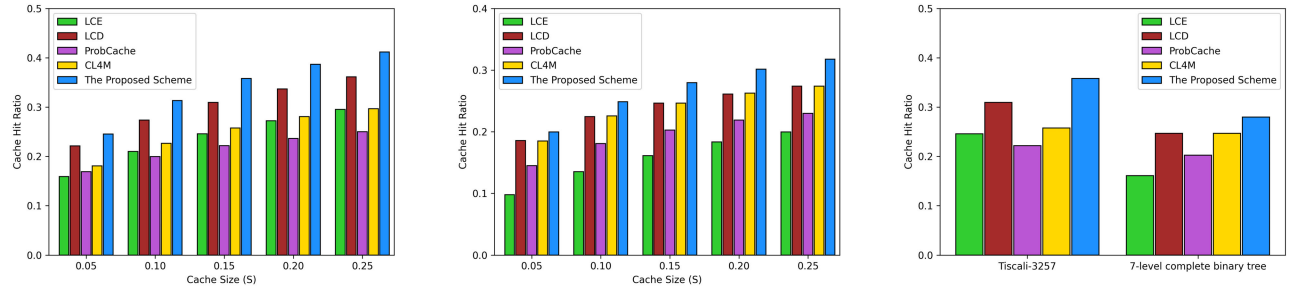


FIGURE 6. Cache hit ratio. (a) Different cache size (S) in Tiscali-3257 topology. (b) Different cache size (S) in tree topology. (c) Different network topologies (S = 0.15, $\alpha = 0.8$).

TABLE 2. Main experimental parameters.

Parameters	Values
Network topology	7-level complete binary tree topology and Tiscali-3257 topology
Number of content objects	100,000 messages
Number of warm up requests	50,000 messages
Number of measured requests	250,000 messages
Content distribution model	Zipf's law, $\alpha \epsilon [0.7,\ 1.1]$
Network caching capacity (S)	S$\in$ [0.05, 0.1, 0.15, 0.2, 0.25]
Link delay	2ms (between the router nodes) 34ms (between router node and provider node)
Content replacement policy	LRU
Number of experiments	5
Number of cycles	30

be more kinds of content, which will lead to a higher cache hit rate. Therefore, the cache hit rate also reflects content redundancy from the side. When the content requested by the consumer is found in a router node, the cache hit will occur. We use the following formula to calculate the cache hit ratio:

$$ACHR = \frac{\sum N_{hit}^{C}}{N_{req}} \tag{8}$$

where N_{hit}^{C} is the number of cache hits for content C, and N_{req} is the total number of requests.

Figs. 6 (a) and 6 (b) show the results of cache hit rate in the two network topologies. The proposed scheme outperforms the other four caching strategies consistently in all the conducted experiments where the parameter S varies across all given values. Fig. 6 (a) shows the results of the cache hit ratio different caching strategies for different parameter S in Tiscali-3257 topology. The cache hit ratio of the proposed scheme is 41.2% when S equals to 0.25, which 5.1% higher than the second-best caching strategy (i.e. the cache hit ratio of LCD is 36.1%). When S $=$ 0.05, the cache hit rate has an improvement by 2.4%, in comparison to LCD. As shown in Fig. 6 (b), in tree topology, the cache hit ratio of the proposed scheme improves by up to 3.1% on average where the S parameter varies ranging between 0.05 and 0.25 in comparison to LCD and CL4M. It is worth noting that the cache hit ratio of CL4M in tree topology is slightly better than that in Tiscali-3257 topology, which is almost the same as that of LCD in tree topology. The improvement of CL4M's cache hit ratio in tree topology happens due to the fact that the average betweenness centrality of nodes in tree topology is larger than that in Tiscali-3257 topology. Moreover, many nodes have the same betweenness centrality in tree topology.

In general, the main reason for the proposed scheme performs well is that it considers popularity reasonably and

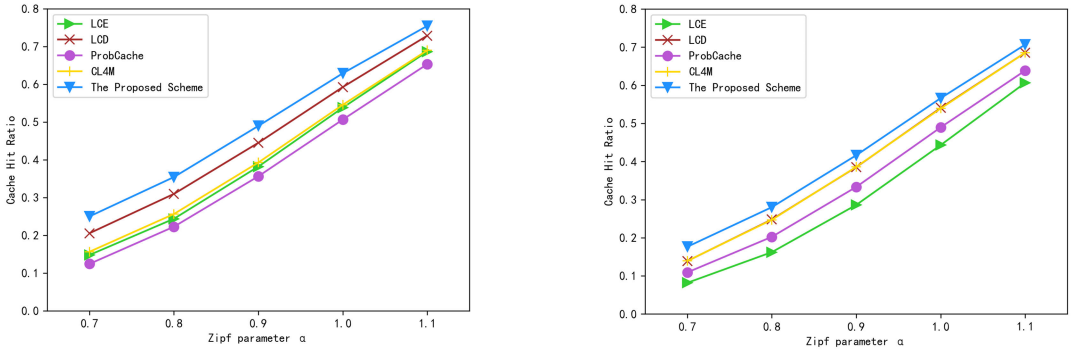


FIGURE 7. Cache hit ratio with different Zipf parameter α (S = 0.15). (a) Tiscali-3257 topology. (b) tree topology.

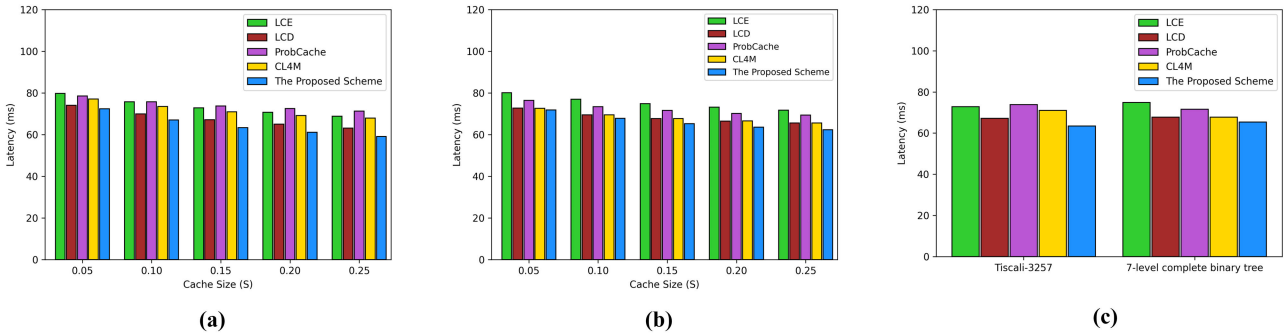


FIGURE 8. Latency. (a) Different cache size (S) in Tiscali-3257 topology. (b) Different cache size (S) in tree topology. (c) Different network topologies (S = 0.15, α = 0.8).

the cache location of the content, thereby improving the utilization of the cache, especially at the edge nodes and the nodes with high popularity in the network. It is also worth mentioning that the performance of the proposed scheme will be significantly improved with increasing S.

For different network topologies, the performance of the same caching strategy is completely different. As shown in Fig. 6 (c), the average cache hit rate of the five cache strategies in Tiscali-3257 topology is higher than that in tree topology obviously with skewness parameter $\alpha = 0.8$ and parameter S = 0.15. The reason for this situation is that nodes in network topologies with high betweenness centrality will be accessed more frequently, which leads to the increase of content replacement times in nodes, thus affecting the performance of the cache hit rate.

The Figs. 7 (a) and 7(b) show the changes in the cache hit rate performance with increasing α and fixed S (where S = 0.15) in Tiscali-3257 topology and tree topology respectively. The cache hit rate of the five cache strategies have an obvious improvement with the increase of parameter α in the two network topologies. This performance change is significantly greater than the performance gain obtained when changing the values of S and fixing the values of α. Consumers are more inclined to request popular content with the increase of Zipf parameter α. The trend of consumer requests is a key factor affecting the cache hit ratio.

B. LATENCY

In addition to the cache hit ratio, latency is an important performance measure in network. A shorter latency can improve the consumer's experience indirectly. The latency of the five caching strategies in Tiscali-3257 topology is shown in Fig. 8 (a). When S is between 0.05 to 0.25, the average latency of the proposed scheme is 64.65 ms, which is 4.8% lower than LCD (67.90 ms), 9.9% lower than CL4M (71.75 ms), 13.1% lower than ProbCache (74.40 ms), and 12.1% lower than LCE (73.58 ms). Similarly, in Fig. 8 (b), the average latency of the proposed scheme (66.2 ms) outperforms the other four caching strategies consistently in tree topology, which is 3.3% lower than LCD (68.45 ms), 3.3% lower than CL4M (68.43 ms), 8.3% lower than ProbCache (72.1 ms), and 12.2% lower than LCE (75.42 ms). In addition, the latency of CL4M in tree topology is almost the same as LCD, which is different from that in Tiscali-3257 topology.

Generally, the latency of the five caching strategies decreases obviously with the increase of cache capacity in the network because more contents could be cached in network. Since the proposed scheme effectively utilizes caching

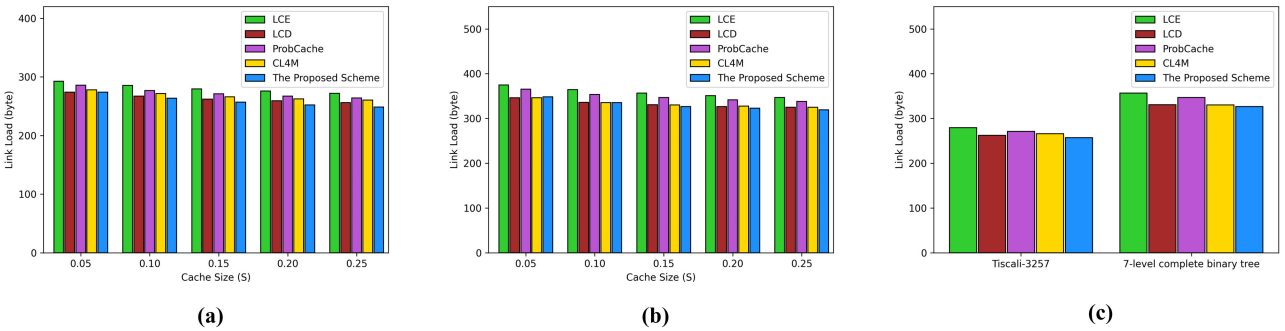


FIGURE 9. Link load. (a) Different cache size (S) in Tiscali-3257 topology. (b) Different cache size (S) in tree topology. (c) Different network topologies ($S = 0.15$, $\alpha = 0.8$).

resources near the consumers, this will incur a performance benefit to other end- consumers in the surrounding network. In other words, LPC cached in edge nodes can satisfy the surrounding consumers to the maximum extent, and GPC is also gradually cached in nodes close to consumers. Therefore, the latency of the proposed scheme performs best in the two network topologies.

As shown in Fig. 8 (c), when skewness parameter $\alpha = 0.8$ and parameter $S = 0.15$, the average latency of the five caching strategies in tree topology is almost the same as that in Tiscali-3257 topology. Under the experimental environment of this article, two different topologies (the tree topology and Tiscali-3257 topology) have little effect on their latency of the caching strategies.

C. LINK LOAD

Figs. 9 (a) and 9 (b) show the link load performance in Tiscali-3257 topology and tree topology against a varying parameter S for a fixed value of skewness parameter α equal to 0.8, respectively. In Tiscali-3257 topology, the proposed scheme performs significantly better compared to the other four caching strategies. The average link load of the proposed scheme in Tiscali-3257 topology for all evaluated scenarios is 259.03 bytes, which is 17.6% better than LCD with an average of 263.67 bytes. Link load reduction happens due to the fact that the proposed scheme caches the content at edge nodes and popular nodes. In particular, when the cache size is small (i.e. $S = 0.05$), the link load of the proposed scheme is 348.26 bytes in tree topology, which is just 0.06% higher than LCD (346.32 bytes). The average link load of the proposed scheme for all evaluated scenarios is 330.35 bytes in tree topology, which is 0.07% better than LCD (332.88 bytes). When S is increased to 0.25, the link load of the proposed scheme improves by 2.9% in Tiscali-3257 topology and 1.8% in tree topology compared with LCD. Overall, the proposed scheme can effectively reduce the link load when the cache size is large. On the contrary, the performance of the link load only has a slight improvement with a small cache size.

As shown in Fig. 9 (c), compared with the Tiscali-3257 topology, there are more consumers in the tree topology. Therefore, the average link load of all of the methods in the tree topology is higher than that in the Tiscali-3257 topology.

VI. CONCLUSION

In-network caching is one of the important features of NDN, allowing content to be cached for a certain time span inside these nodes to fulfill the requirements of subsequent interests. As we know, caching has the ability to improve the overall performance of the NDN architecture. In this article we propose a new mechanism for managing the transformation of the content in a network to resolve the previous issues of other algorithms in NDN. The proposed scheme divides content popularity into different types for discussion and calculates the different types of content popularity in a lightweight way. In addition to content popularity, the proposed scheme also considers node popularity to further improve the utilization rate of a single node. Furthermore, the proposed scheme obtains a reasonable cache location for the content that needs to be cached by analyzing the characteristics of the transmission path in the topology. To evaluate the proposed mechanism, we performed a simulation campaign in two different network topologies by using Icarus simulator. The results show that the proposed scheme in terms of the cache hit ratio, start latency, and link load performed much better than all other strategies (LCD, LCE, CL4M, and ProbCache).

For future research, we will discuss the fairness of the proposed scheme and propose a more effective cache replacement strategy to replace the LRU according to the characteristics of cache contents in nodes.

REFERENCES

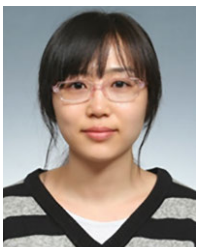
[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[2] Cisco. (2018). *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. V. N. I. Accessed: Aug 15, 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collate-ral/serviceprovider/visual-networking-index-vni/white-paper-c11-741490.html

[3] S. Akhshabi and C. Dovrolis, "The evolution of layered protocol stacks leads to an hourglass-shaped architecture," in *Proc. ACM SIGCOMM Conf. SIGCOMM (SIGCOMM)*, 2011, vol. 41, no. 4, pp. 206–217.

[4] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, "A survey on content-oriented networking for efficient content delivery," *IEEE Commun. Mag.*, vol. 49, no. 3, pp. 121–127, Mar. 2011.

[5] R. Steinmetz and K. Wehrle, "Peer-to-peer-networking &-computing," *Informatik Spektrum.*, vol. 27, no. 1, pp. 51–54, Feb. 2004.
[6] P. Agyapong and M. Sirbu, "Economic incentives in information-centric networking: Implications for protocol design and public policy," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 18–26, Dec. 2012.
[7] M. Amadeo, C. Campolo, and A. Molinaro, "NDNe: Enhancing named data networking to support cloudification at the edge," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2264–2267, Nov. 2016.
[8] M. Wang, J. Wu, G. Li, J. Li, and Q. Li, "Fog computing based content-aware taxonomy for caching optimization in information-centric networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, May 2017, pp. 474–475.
[9] M. Amadeo, C. Campolo, G. Ruggeri, G. Lia, and A. Molinaro, "Caching transient contents in vehicular named data networking: A performance analysis," *Sensors*, vol. 20, no. 7, pp. 1–17, Apr. 2020.
[10] M. Naeem, R. Ali, B.-S. Kim, S. Nor, and S. Hassan, "A periodic caching strategy solution for the smart city in information-centric Internet of Things," *Sustainability*, vol. 10, no. 7, p. 2576, Jul. 2018.
[11] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Netw.*, vol. 29, no. 3, pp. 68–74, May 2015.
[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerg. Netw. Experiments Technol. (CoNEXT)*, 2009, pp. 1–12.
[13] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Comput. Netw.*, vol. 57, no. 16, pp. 3128–3141, Nov. 2013.
[14] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical Web caches," in *Proc. IEEE Int. Conf. Perform., Comput., Commun.*, Phoenix, AZ, USA, 2004, pp. 445–452.
[15] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Perform. Eval.*, vol. 63, no. 7, pp. 609–634, Jul. 2006.
[16] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. 2nd Ed. ICN Workshop Inf.-Centric Netw. (ICN)*, 2012, pp. 55–60.
[17] K. Thar, S. Ullah, R. Haw, T. LeAnh, T. Z. Oo, and C. S. Hong, "Hybrid caching and requests forwarding in information centric networking," in *Proc. 17th Asia–Pacific Netw. Operations Manage. Symp. (APNOMS)*, Busan, South Korea, Aug. 2015, pp. 203–208.
[18] J. H. Mun and H. Lim, "Cache sharing using Bloom filters in named data networking," *J. Netw. Comput. Appl.*, vol. 90, pp. 74–82, Jul. 2017.
[19] F. M. Modesto and A. Boukerche, "An analysis of caching in information-centric vehicular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.
[20] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," in *Proc. 21st Int. Conf. Comput. Commun. Netw. (ICCCN)*, Munich, Germany, Jul. 2012, pp. 1–7.
[21] Y. Wang, K. Lee, B. Venkataraman, R. L. Shamanna, I. Rhee, and S. Yang, "Advertising cached contents in the control plane: Necessity and feasibility," in *Proc. IEEE INFOCOM Workshops*, Orlando, FL, USA, Mar. 2012, pp. 286–291.
[22] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proc. IEEE INFOCOM Workshops*, Orlando, FL, USA, Mar. 2012, pp. 316–321.
[23] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in named data networking," in *Proc. 8th ACM/IEEE Symp. Architectures Netw. Commun. Syst. (ANCS)*, 2012, pp. 15–26.
[24] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache 'less for more' in information-centric networks (extended version)," *Comput. Commun.*, vol. 36, no. 7, pp. 758–770, Apr. 2013.
[25] J. Ren, W. Qi, C. Westphal, J. Wang, K. Lu, S. Liu, and S. Wang, "MAGIC: A distributed MAx-gain in-network caching strategy in information-centric networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Apr. 2014, pp. 470–475.
[26] C. Bernardini, T. Silverston, and O. Festor, "A comparison of caching strategies for content centric networking," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.
[27] Q. N. Nguyen, J. Liu, Z. Pan, I. Benkacem, T. Tsuda, T. Taleb, S. Shimamoto, and T. Sato, "PPCS: A progressive popularity-aware caching scheme for edge-based cache redundancy avoidance in information-centric networks," *Sensors.*, vol. 19, no. 3, pp. 1–18, Feb. 2019.
[28] V. Sourlas, I. Psaras, L. Saino, and G. Pavlou, "Efficient hash-routing and domain clustering techniques for information-centric networks," *Comput. Netw.*, vol. 103, pp. 67–83, Jul. 2016.
[29] S. Ullah, K. Thar, and C. S. Hong, "Management of scalable video streaming in information centric networking," *Multimedia Tools Appl.*, vol. 76, no. 20, pp. 21519–21546, Oct. 2017.
[30] Z. Zhang, C.-H. Lung, M. St-Hilaire, and I. Lambadaris, "Smart proactive caching: Empower the video delivery for autonomous vehicles in ICN-based networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7955–7965, Jul. 2020.
[31] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, Jun. 2012.
[32] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. IEEE Conf. Comput. Commun., 18th Annu. Joint Conf. IEEE Comput. Commun. Societies Future (INFOCOM)*, New York, NY, USA, 1999, pp. 126–134.
[33] H. J. Stuart, "The exponentially weighted moving average," *J. Quality Technol.*, vol. 18, no. 4, pp. 203–210, 1986.
[34] D. C. Montgomery, L. A. Johnson, and J. S. Gardiner, *Forecasting and Time Series Analysis*. New York, NY, USA: McGraw-Hill, 1990, p. 381.
[35] L. Saino, I. Psaras, and G. Pavlou, "Icarus: A caching simulator for information centric networking (ICN)," in *Proc. 7th Int. Conf. Simul. Tools Techn.*, 2014, pp. 66–75.
[36] B. Zhang, T. S. Eugene Ng, A. Nandi, R. H. Riedi, P. Druschel and G. Wang, "Measurement based analysis, modeling, and synthesis of the internet delay space," in *Proc. 6th ACM SIGCOMM Conf. Internet Meas.*, 2006, pp. 85–98.
[37] J. Rajahalme, M. Särelä, K. Visala, and J. Riihijärvi, "On name-based inter-domain routing," *Comput. Netw.*, vol. 55, no. 4, pp. 975–986, Mar. 2011.



YIQI GUI received the B.S. degree in computer science from Beihua University, China, in 2004, and the M.S. and Ph.D. degrees in computer information and communication engineering from Kangwon National University, South Korea, in 2007 and 2012, respectively.

From 2007 to 2009, she was a Research Assistant with the Database and Multimedia Laboratory. Since 2012, she has been an Assistant Professor with the Computer Science Department, Yangzhou University, China. She was a Visiting Scholar with Kangwon National University, in 2017. Her research interests include peer to peer multimedia systems, multimedia in information-centric networks and future networks, and network security. She is a member of CCF and a reviewer of several SCI-indexed journals and international conferences.

YONGKANG CHEN received the B.S. degree in computer science from Nantong University, China, in 2016. He is currently pursuing the M.S. degree with Yangzhou University, China.

His research interests include data mining for multimedia systems and network caching technology for information-centric networking (ICN).