A Pre-Caching Strategy Based on the Content Relevance of Smart Device’s Request in Information-Centric IoT

MOLI ZHANG, BOWEI HAO, RUILIN WANG, AND YE WANG
School of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China
Corresponding author: Bowei Hao (180318050220@stu.haust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 61971458, and in part by the basic research projects in the University of Henan Province under Grants 19zx010.

ABSTRACT In recent years, investigations on various forms of solution for Internet of Things (IoT) paradigm opens the new era of smart devices. The Information-Centric Internet of Things (IC-IoT) can apply the characteristics of content caching which is a typical feature of Information-Centric Networking. Therefore, designing an efficient caching strategy for IC-IoT is able to improve the efficiency of smart devices. In this paper, we propose a Pre-Caching Strategy based on the Relevance of smart device request Content, namely PCSRC. After the first request of content chunk at a smart device side, PCSRC pre-caches the rest of the content chunks. There are two types of cached content chunks, that is, actual requested content chunks and pre-caching content chunks. Actual requested content chunks are pushed forward hop-by-hop according to the trend of the local activity. The popular content will be gradually pushed to the edge of IC-IoT. Furthermore, a sojourn time of a cached content chunk is employed to save storage space. The simulation results show PCSRC can reach 42.7% on the cache hit ratio with the different storage sizes. For the average hop count, PCSRC can increase about 42.2% than that of ProbCache. And PCSRC can improve about 47.9% than that of CEE for the average hop count.

INDEX TERMS Information-centric networking, Internet of Things, content chunk, local activity, pre-caching.

I. INTRODUCTION

With the rapid development of the Internet of Things (IoT), this paradigm opens a new era of smart devices and has become a comprehensive application of a new generation for information technology [1], [2]. On traditional networks, researchers have studied different research directions of the IoT, such as routing [3], buffering [4], privacy [5], computing offloading [6], perception and fusion [7], [8], identification [9], and optimization [10], etc., to improve the overall performance of the IoT. However, in the traditional network, it has a huge challenge due to the fundamental problems of the network architecture, for example, decoupling location and information and the node cannot caching content in-network [11]–[13]. Therefore, Information-Centric Networking (ICN) is proposed as one of the popular network architecture for future Internet [14]. As the most promising future network architecture, ICN decouples information content from location, users can request through content names, and ICN also offers clear benefit that the network performance is improved by increasing the effect of the caching in-network [4], [15]. Information-Centric Internet of Things (IC-IoT) can apply the ICN’s the characteristics of content caching to the IoT, which can reduce the energy loss and improve the performance of the network.

Due to the characteristics of IC-IoT caching content in-network, traditional caching algorithms for Web, CDN and P2P cannot be directly applied. In recent years, many researchers have studied and explored the cache in ICN and obtained many research achievements. However, there are still some issues with improving network performance by caching content. Currently, those issues can be attributed to the following two aspects, namely, selecting which content to cache or selecting which nodes to place the content.
The classic design of caching strategy for ICN is Cache Everything Everywhere (CEE) [16]. However, because of blind cached contents, some cached contents tend to be homogeneous. When a user requests a content, most of the requests are still forwarded to the content server, which increases the request delay and wastes storage. In order to improve the efficiency of the caching system, the researchers have launched a series of works [17]–[19]. Zhang et al. [18] have taken the popularity of contents as a basis for content caching. However, the connection between content requests is not considered. Taking the content chunk as a starting point, a collaborative caching algorithm is proposed to reduce the request delay [19]. When a content is divided into chunks, the user will request subsequent content chunks after requesting a certain content chunk. The above work has verified that introducing the relevance of the requested content into the caching system will greatly help improve the performance of the network. Therefore, in IC-IoT, it is important to study a pre-caching strategy based on the content relevance of smart device requests to improve the performance of the network. Nevertheless, there are few studies on the combination of the content popularity and the relevance to content requests.

In response to these deficiencies, PCSRC is proposed based on the relevance of smart device’s requests. IC-IoT uses the characteristics of ICN, where the request and return of the content follows the same path. In IC-IoT, when a smart device initiates a request for a content chunk, PCSRC caches the subsequent content chunks that follow the content chunk of smart device’s request and the smart device is the user. Therefore, there are two types of content in the network, which are the actual request content and the pre-caching content, respectively. In PCSRC, we set the router ID list field in an interest packet to obtain the request path. Then the pre-caching content is cached by the router’s ID list of the interest packet. In order to fully exploit the cache space, we cache the content in the second half of the path. With the increase of the popularity of request contents, the popular contents are advanced to edge nodes. Furthermore, in order to save storage space, we employ the sojourn time for cached content.

The main contributions in this paper are as follows:

1. We introduce a pre-caching strategy which can pre-cache the subsequent content chunks of a requested content. In PCSRC, cached content chunk will be pushed to the edge of IC-IoT hop-by-hop based on its local activity. Moreover, we employ sojourn time of content chunks to reduce the consumption of spaces.

2. We perform extensive evaluations to validate PCSRC in comparison to CEE and ProbCache [17], respectively. The simulation results show that PCSRC effectively reduces the number of hops and access delays. In addition, PCSRC achieves relatively high hit rate than other two strategies.

The rest of this paper is organized as follows. In Section II, we give related works. In Section III, we propose a pre-caching strategy based on the relevance of requested content. In Section IV, we evaluate the proposed caching strategy. In Section V, we conclude this paper.

II. RELATED WORK

In IC-IoT, the ability of traditional network nodes for forwarding and routing to give the node cache of the ICN network. On the basis of traditional web caching [20] and proxy caching [21], an increasing number of cache strategies have been proposed in ICN to improve the efficiency of caching and the performance of the network.

ICN usually uses the CEE-based scheme to decide cache locations [16]. This scheme does not make any choice to cache location and content, so as to cache a lot of useless content, resulting in redundant content storage and difficulty to exert the purpose of cache adequately. Laoutaris et al. [20] proposed a cache scheme Leave Copy Down (LCD) to improve the utilization of cache space. When a request hits a cached content, it is cached only at the next direct hop of the hit node to reduce the redundancy of the cached content. A scheme of Move Copy Down (MCD) is also proposed in [20]. Unlike LCD, when a cache hits, the content of the current node is deleted in addition to the content cached to the downstream node. The above methods are to reduce the redundancy of content by limiting the number of copies of the same content along the path. However, as networks become more and more complicated, the performance of LCD and MCD may significantly reduce. A cache mechanism based on probability is proposed in [17]. This cooperative network caching is based on the path lengths called ProbCache that approximates the capability of paths to implement the content caching. The caching strategy ProbCache can effectively reduce redundant data. Chai et al. [22] proposed a selective cache mechanism based on betweenness. They calculate the betweenness of nodes along the path, and then select the node with the largest betweenness as the content cache node to reduce the number of replacements and content redundancy. However, this scheme will result in frequent replacement of the cache on a few important nodes.

In the aspect of decisions with cache content, Wang et al. [23] proposed an alternative algorithm called LB (Least Benefit) that considers both content access frequency and cache distance benefit. In particular, the current gain is counted into the total gain for each cache hit. This algorithm considers the single cache gain and access frequency at the same time. Meanwhile, additional overhead is added due to the need to maintain the gain information. In [24], the optimal content distribution and storage capacity are considered based on the integer linear programming formulation. Mangili et al. [25] proposed a centralized strategy to combine storage available on wireless access points with lease the unused bandwidth. A cooperative caching strategy based on popularity rank of content is proposed in [26]. According to the number of requests, the number of cached data is gradually increased in a way of exponential growth. However, the scheme does not take into account the relevant requests.
of subsequent users, but only satisfied with the differentiated cache on spatial cache locations.

In the decision of cache location and cache content, Ming et al. [27] proposed an ideological strategy based on the lifecycle of the content. The content could be replaced only when the lifecycle times out. The lifetime of a content is determined by its storage location and popularity. That is, the closer a content is to the edge of the network, the higher its popularity is, and the longer it lifecycle is. The content package carries the lifecycle field which is adjusted by the router along the path to determine the lifecycle of the content. However, this scheme assumes that the popularity of content is known as a static parameter, which may not reflect the dynamic evolution characteristics of the content request and the parameter is difficult to pre-know. Ren et al. [28] proposed a cache strategy for maximum revenue. The strategy determines the specific cache location by calculating the revenue of the cached content on the request path. On the basis of this, Wu et al. [29] proposed a probability caching strategy based on maximizing cache revenue. Considering the function of popularity is predetermined in the design of the caching algorithm, which may not reflect the dynamic variability and the correlation characteristics of the request distribution. In order to solve the above limitations, we propose a pre-caching mechanism PCSRC, where each node cooperates with others to pre-cache the content which may be requested later to avoid congestion [30]. According to the local activity of the content chunk, it promotes the marginalization of the popular content.

III. A PRE-CACHING STRATEGY BASED ON THE RELEVANCE OF REQUESTED CONTENT

Our network model is defined as an undirected graph \( G(V, E) \), where \( V = \{v_1, v_2, \ldots, v_w\} \) is a collection of router nodes. The subscript \( w \) denotes the number of network nodes. \( E = \{e_{ij} = (v_i, v_j)|v_i, v_j \in V\} \) represents a set of edges between pairs of nodes, where each \( e_{ij} \) represents a bidirectional communication links between \( v_i \) and \( v_j \). Let \( O = \{O_1, O_2, \ldots, O_k, \ldots, O_M\} \) be a set of content objects, where \( M \) represents the number of content in a network. Each content is divided into \( n_k \) content chunks, namely \( O_k = \{O_k,1, O_k,2, \ldots, O_k,i, \ldots, O_k,n_k\} \), i.e.,

\[
|O_k| = \sum_{i=1}^{n_k} |O_k,i|, \quad (1)
\]

where \( |\cdot| \) is the size of a content chunk. In this paper, the name of a content chunk is made up of the content name and the relative position of the content chunk. The name of a content chunk is the only credential which is used to distinguish from other chunks. Assume that there is enough available bandwidth to support forwarding content packets. The appropriate number of chunks for a content and the scalability of partitioning a content are not within the scope of this paper.

A. THE LOCAL ACTIVITY OF THE CONTENT CHUNK

In order to reflect the dynamic and limitation of content requests, the sliding window mechanism is adopted. This mechanism is dynamically calculated and updated the local activity of a content chunk, which considers both the hot degree of historical requests and the novelty of current requests.

Definition (Local Activity of a Content Chunk (LACC)): We use \( t_{x+1} \) to denote the time which is the \((x+1)\)-th request for a content \( O_{k,i} \). The number of requests in the sliding window at the moment \( t_{x+1} \) is called the local activity of a content chunk, i.e.,

\[
A_{O_{k,i}} = \sum_{j=1}^{K} \lambda_{O_{k,i},j} + \lambda_{O_{k,i},(t_{x+1})}, \quad (2)
\]

where parameter \( K \) is the size of the sliding window and the time window consists of \( K \) time slots of length \( T \). \( \lambda_{O_{k,i},j} \) represents the number of accessing to content \( O_{k,i} \) in the \( j \)-th time slot \( T_j \) of sliding window \( ((j - 1) \times T + j \times T) \), namely instant popularity in the time slot \( T_j \). \( \lambda_{O_{k,i},(t_{x+1})} \) represents the number of requesting content \( O_{k,i} \) in the time slot \(((K - 1)T + t_{x+1})\).

As shown in Figure 1, the dynamic update of LACC is driven by requests. The activity of a content is updated only when the current request arrives.

**FIGURE 1.** The local activity of a content chunk.

Formula (2) does not take the novelty of content chunks into account. In other words, the frequency of request has different impacts on LACC in different time slot of sliding window. Therefore, we introduce the weight parameters \( \gamma \), and then have

\[
A_{O_{k,i},(t_{x+1})} = \sum_{j=1}^{K} \frac{1}{\gamma^{K-j}} \lambda_{O_{k,i},j} + \lambda_{O_{k,i},(t_{x+1})}, \quad (3)
\]

where \( \gamma > 1 \). We infer from the formula (3) that the previous slot in the sliding window is less important to the LACC, while the newer slot has significant effects on LACC. This weight parameter reduces the impact of history requests to LACC. It not only ensures the content’s history request information has impact on the current information but also meets...
the novelty of the content request, which reflects the current result of LACC more accurately.

B. IC-IoT PACKET STRUCTURE

In order to distinguish initiated packets by user from pre-cache packets, we refer to them as interest packets, pre-cache interest packets, data packets and pre-cache data packets, respectively. The specific format of packet is shown in Figure 2.

![FIGURE 2. The packet formats.](image_url)

Pre-cache interest packets and pre-cache data packets are used to support operations of pre-caching. The subsequent content chunks are encapsulated in pre-cache data packets. For example, a user requests the content \( O_{k,i} \), afterwards the router nodes will cache the content chunks \( O_{k,i+1} \ldots O_{k,n_i} \). When a content chunk is first request, implies that the content chunk must be transferred from the the Content Source Server (CSS) to the user on request. However, the content chunk is close to the user with increasing request. The hop count and interest packet generation are reduced. So the network overhead is will also reduce. In order to distinguish pre-cache content chunks and request content chunks, we add the Packet Type field in the packet. The field can effectively control the invalid forwarding of the cache packet to reduce the work process of subsequent routers.

If there are no new interest packets arriving to a node, the node will delete the corresponding entry from it PIT quickly, so the pre-cache effect is not achieved. Therefore, we introduce synthetic interest generator [31] to generate pre-cache interest packets. If a pre-cached content chunk has been placed on the path \( l_{io} = (e_{i,j+1}, e_{i+1,j+2}, \ldots) \), the CSS will not receive the interest packet, and each router in the path will not repeatedly cache the pre-cache data packet. \( l_{io} \) represents the path from edge router \( v_i \) to CSS.

There are two problems when implementing above ideas: (1) how to perceive a path of interest packets passing in order that data packets or pre-cache data packets are cached on the path \( l_{io} \); (2) the pre-cached or cached content chunks will occupy the storage space of CS. Hence how to reasonably set a sojourn time for content chunks in CS is a problem.

![Algorithm 1 A Node Processes Data Stream Algorithm](algorithm_url)

To solve the first problem, we add a Router ID List (RIL) field to interest packet. The RIL field represents a collection of ID of routers. Each router adds its ID into RIL when it forwards the interest packet toward the upstream node. In addition, we add a Cache Node Identifier (CNI) to the selector field. CNI was used to identify where a content chunk will be cached. The CNI field has two roles: (1) CS will cache (pre-cache) the content to the corresponding router node, where CNI corresponds to the router ID on the path \( l_{io} \). We further illuminate the pre-cached strategy by using the Eq.(4) and Eq.(5). (2) the field is set to 0 or \(-1\) by capturing the change of popularity when users request a content chunk. When the next hop node receives a packet, the value of this field will be checked. If the value of CNI field is 0, the router will cache the content chunk. If the value of CNI field is \(-1\), the node will forward the content chunk. In order to solve the second problem, we add the LACC field. LACC represents the local activity of the content chunk. It was used in the calculation of sojourn time and the downward movement of storage location.

The process that a node deals with a data stream is shown in Figure 3. When a router node \( v_i \) receives a data packet and a pre-cache data packet about content chunks \( O_{k,i} \ldots O_{k,n_i} \), it first checks its PIT. If there is no entry in the PIT, the node \( v_i \) will discard the packet about content chunks \( O_{k,i} \ldots O_{k,n_i} \). Otherwise, it first checks the value of the Packet Type field.
If the packet is a data packet and the value of the CNI field is 0, node $v_i$ will cache the content chunk and set its sojourn time based on LACC. If the value of CNI matches with the ID of node $v_i$, $v_i$ will cache the content chunks $O_{k,i},...,O_{k,n}$, set the sojourn time $T_{basic}$ and forward the data packet. If the value of CNI does not match with the ID of node $v_i$, it will forward the packet directly. If the packet is a pre-cache data packet and the ID of node $v_i$ matches with the value of CNI, it will cache the chunk and set its sojourn time $T_{tem}$. Otherwise, it forwards the content chunk only. The algorithm is shown in Algorithm 1.

The process that a node deals with an interest packet is shown in Figure 4, where we assume that there is enough available bandwidth to support forwarding content packets. Packets generated in label 1 are interest packets, generated in label 2 are pre-cache interest packets. First, when a node receives an interest packet, it responds to the interest packet directly if the content exists in its CS, where the node will calculate $A_{O_{k,i}}(t_{x+1})$ to decide the value of CNI. Second, if the content does not exist in its CS, the node will check whether there is a request record in its PIT. If the request record exists in PIT already, the request port will be added to the corresponding entry of its PIT. Otherwise, if there is a corresponding entry in its FIB, it will add the record of forwarding to its PIT, sent the synthetic interest of subsequent content chunks, and forward the content chunk to the next hop. If there is a corresponding entry in its FIB, the interest packet will be discarded directly. The process is shown in Algorithm 2.

C. A COOPERATIVE PRE-CACHING ALGORITHM

If a user wants to get complete information, a request for different chunks of the same content will be continuously issued after the content is divided into many content chunks. After a specific user requests a certain content $O_{k,i}$, if the $i$-th chunk of the content $O_{k,i}$ was also pre-cached on the path $l_{io}$, it will inevitably reduce the response time that the user requests the remaining content chunks and improve the possibility that the user obtains the required content chunk nearby. If a user sends an interest packet, the CSS will send subsequent content chunks to the user. When the router node receives a data packet or a pre-cache data packet, it will cache or pre-cache the content chunk according to the RIL in the interest packet.
Algorithm 2 A Node Processes Interest Stream Algorithm
1: The node receive interest packet;
2: if the item is in the CS then
3: Calculate the LACC;
4: if \( A_{O_k,i}(t_{x+1}) \geq A_{O_k,i}(t_x) \) then
5: Set CNI = 0;
6: Returning Data;
7: else
8: Set CNI = -1;
9: Returning Data;
10: end if
11: else
12: if the request exists in the PIT then
13: Add the request port; Drop the interest packet;
14: else
15: Search the FIB; Get the forwarding port and add entry in the PIT;
16: Forward the Pre-Cache interest packet;
17: end if
18: end if

1) THE STORAGE DECISIONS OF PRE-CACHING
CSS sends subsequent chunks of the content about user requests to the designated router node according to interest packets. We assume that the request content \( O_k \) contains \( n_k \) chunks, the current content chunk requested by a user is \( Q_{k,m} \), and the number of hops from the user to the CSS is \( h \). Hence, the number of the subsequent content chunks of requested content is \( n_k \). For each router, the number of cached contents is \( f \) for the path \( l_{io} \), i.e.,

\[
f = \left\lceil \frac{(n_k - m + 1)}{h/2} \right\rceil = \left\lceil \frac{2(n_k - m + 1)}{h} \right\rceil. \tag{4}
\]

In order to make full use of edge nodes, the pre-cached content is placed in the half path of \( l_{io} \). The corresponding relationship between the storage of content and the node is designed as follows,

\[
q = \left\lfloor \frac{h}{2} \right\rfloor + \left\lceil \frac{(j - m + 1)}{f} \right\rceil. \tag{5}
\]

where \( q \) represents the subscript of a router, that is, the content chunk \( O_{k,j} \) is send to the node \( v_q \) for caching; \( j \) represents the \( j \)-th content chunk of the content \( O_k \) whose initial value is the subscript of the requested content chunk \( (j \geq m) \). For example, a user requests a content chunk \( O_{k,m} \), and the initial value of \( j \) is set to \( m \). The caching algorithm of PCSRC is shown in Algorithm 3.

2) THE PUSHING OF THE CACHE CONTENT
In PCSRC, the popular content chunks are pushed into the edge of the network based on the LACC. When a cached node receives an interest packet \( req_{Q_{k,j}} \), it computes LACC of the content chunk \( Q_{k,j} \).

1) If \( A_{O_k,i}(t_{x+1}) \geq A_{O_k,i}(t_x) \), it represents the popularity of content increases. The node \( v_i \) sets the value of CNI in corresponding packet of \( req_{Q_{k,j}} \) to 0. When a downstream node \( v_{i-1} \) receives the packet, it will check CNI. If the value of CNI is 0, it represents that the content needs to be cached. Then \( v_{i-1} \) will calculate the sojourn time based on LACC to move the content to the next-hop. Since \( v_{i-1} \) cached the content, \( v_i \) will not receive the packet of content \( Q_{k,j} \). When the sojourn time expires, the content will be in the “deletable” state. Otherwise, \( v_{i-1} \) will set the CNI field to -1 after \( v_{i-1} \) caches the content, which prevents downstream nodes to store the content again.

2) If \( A_{O_k,i}(t_{x+1}) < A_{O_k,i}(t_x) \), it represents the popularity of content decreases. The node \( v_i \) will adjusted the sojourn time of content chunks according to LACC, and then respond to downstream nodes by sending the packet to them. When downstream nodes receive the packet, they will check CNI to decide whether the content chunk needs to be cached. Because the value of CNI is -1, the node only needs to forward the content chunk.

3) When the node receives the pre-cache interest packet, it shows that the packet is a request of the content chunk generated by user association. CSS sends the pre-cache data packet of the remaining content chunk. After the router corresponding to the CNI receives the pre-cache packet, this node will extract the pre-cached content chunks and set the temporary sojourn time. The content will be stored in CS in order to facilitate the requests of subsequent content chunks of user.

3) THE CALCULATION OF SOJOURN TIME
According to the type of packet, we have to set the stay time in separate situations. For data packets requested by the user, we dynamically change the sojourn time according to the sliding window. For pre-cached content chunks, we set a temporary sojourn time to prevent the pre-cached content from taking up longer storage space.

a: THE SOJOURN TIME OF THE REQUEST CONTENT CHUNK
The value of sojourn time is dynamically calculated according to LACC. The larger the value of LACC is, the longer the sojourn time is. It aims to increase the sojourn time and the subsequent utilization of popular content chunks. If \( A_{O_k,i}(t_{x+1}) < A_{O_k,i}(t_x) \), the sojourn time is updated directly

Algorithm 3 Pre-Caching Algorithm
1: get the path that the interest has been through \( l_{io} \), where \( v_i \) is the closest to the consumer;
2: Interest \( req_{Q_{k,j}} \) for chunk \( Q_{k,m} \) from users;
3: \( f \leftarrow \lceil (2n_k - m + 1) / h \rceil; \)
4: \( j = m; \)
5: while \( j \leq n_k \) do
6: \( q \leftarrow \lceil h/2 \rceil + \lceil (j - k + 1)/f \rceil; \)
7: send \( O_{k,j} \) to \( v_q \);
8: \( j++; \)
9: end while
according to LACC. If \( A_{O,k}(t_{k+1}) \geq A_{O,k}(t_k) \), the value of CNI field is set to 0, which indicates that the downstream node needs to cache the content chunk and calculate the sojourn time.

If a user requests a content chunk for the first time (LACC = 1), there is no any intermediate node with the cached content chunk. Hence the request is sent to CSS, and then CSS caches the content chunk according to the decision of the content chunk’s storage. The sojourn time is set to the basic sojourn time \( T_{\text{basic}} \). After that, if a user requests this content chunk again, its sojourn time is updated dynamically according to LACC, i.e.,

\[
ST(x + 1) = ST(x) + \left(1 - \frac{1}{A_{O,k}(t_{k+1})}\right) \cdot T_{\text{basic}},
\]

where \( ST \) denotes the sojourn time. From the formula (6), the value of the sojourn time increases with the increasing of the value of LACC. It means that the greater the popularity is, the longer the sojourn time is.

\[b: \text{THE PRE-CACHED SOJOURN TIME}\]

Considering that a user may send a request for different content chunks of the same content within a short time, we propose a pre-caching strategy for content chunks to reduce the delay of requesting subsequent content chunks. We set \( PST = T_{\text{temp}} \) for pre-cached content chunks to prevent them from taking up longer storage space of CS. The value of \( PST \) of a pre-cached content chunk is set by the average sending time slot of content requests, which has no relationship with the actual number of requests or the activity of content. That is to say, the value of \( PST \) of a pre-cached content chunk is set according to the interval of sequential requests. When a user requests sequential contents, \( PST \) updates dynamically according to LACC. If the pre-cached content is requested firstly, LACC is set to 1. According to the policy of storage time, \( PST(1) = T_{\text{basic}} \).

\[4) \text{CACHE REPLACEMENT POLICY}\]

When the storage space of a node is full, it needs to delete some cached content chunk to cache the latest content. For a pre-cache content chunk, if it is not requested by users for a period of time, it quickly becomes a \textit{deletable} state. When a new content arrives, the content chunk in \textit{deletable} state will be replaced first. If there is no any content chunk in \textit{deletable} state, the new content will replace the content chunk with the minimum \( ST \) in the CS.

\[5) \text{AN EXAMPLES FOR PCSRC}\]

Figure 5 shows a specific operation instance of PCSRC algorithm. Assume that CSS contains content chunks \( O_{k,1} \sim O_{k,5} \). \( R_1 \sim R_3 \) represent users. \( req^{O_{k,1}} \) represents the request that a user sends a request for content \( O_{k,1} \).

(1) A user sends an interest packet \( req^{O_{k,1}} \) for the content chunk \( O_{k,1} \). Since the content chunk is requested for the first time, it is not cached in the path. Hence, \( req^{O_{k,1}} \) is sent to CSS and records the router ID along the path. When the CSS receives \( req^{O_{k,1}} \), it extracts the RIL which contains the ID of the node \( v_1, v_2, v_3, v_4 \). After that, the CSS will prepare for pre-caching. If we suppose a router can cache 3 content chunks at most, \( O_{1,1} \sim O_{1,3} \) are cached in the node \( v_3 \) and \( O_{1,4} \sim O_{1,5} \) are cached in the node \( v_4 \). The node corresponding to this caching strategy will be gained by formula (4) (5). This strategy can uniformly cache \( O_{k,1} \sim O_{k,5} \) on the second half of the path. When the node \( v_1 \sim v_4 \) receives the data packet, it will check the CNI. If CNI matches oneself, the node will cache the content chunk. Hence, \( v_3 \) caches \( O_{1,1} \sim O_{1,3} \) and \( v_4 \) caches \( O_{1,4} \sim O_{1,5} \). In addition, the node decides whether to forward the content to the next node depending on the Packet Type field. When the node \( v_3 \) receives the data packet with \( O_{k,1} \), it will check the Packet Type field and the LACC to decide whether to forward and set the sojourn time. After that it sets the CNI field to \(-1\), and forwards the data packet. When the node \( v_2 \) and \( v_1 \) receive the data packet, they will directly forward the packet if the CNI field is \(-1\). This process is shown in Figure 5 (1).

(2) User \( R_2 \) sends interest packets \( req^{O_{k,1}} \sim req^{O_{k,3}} \) to request content chunks \( O_{k,1} \sim O_{k,3} \). When node \( v_1 \) receives an interest packet, it forwards the interest packet to the upstream node \( v_2 \) since it does not cache the content chunks \( O_{k,1} \sim O_{k,3} \). Similarly, \( v_2 \) forwards the interest packet to \( v_3 \). When node \( v_3 \) calculates the LACC. Due to \( A_{O_{k,1}}(2) \geq A_{O_{k,1}}(1), A_{O_{k,2}}(2) \geq A_{O_{k,2}}(1) \) and \( A_{O_{k,3}}(2) \geq A_{O_{k,3}}(1) \), \( v_3 \) sets CNI of the data packet to 0 to respond to the request. When node \( v_2 \) receives the data packet, it will check the CNI, calculate the \( ST \), store the content in the CS, set CNI to \(-1\), and forward the data packet. After a period of time, since the node \( v_3 \) has not received the request, the content chunks \( O_{k,1} \sim O_{k,3} \) will be in \textit{deletable} state. This process is shown in Figure 5 (2).

(3) In case of (2), suppose that the activity of \( O_{k,3} \) decreases in this request, namely \( A_{O_{k,3}}(2) < A_{O_{k,3}}(1) \). When the content chunks \( O_{k,1} \) and \( O_{k,2} \) in \( v_3 \) is expired, the data packet of \( O_{k,3} \) is not cached in the next hop. Moreover, the CNI of the packet of \( O_{k,3} \) is set to \(-1\), and the \( ST \) of the packet is calculated based on the LACC. This process is shown in Figure 5 (3).

\[IV. \text{PERFORMANCE EVALUATION}\]

\[A. \text{SIMULATION ENVIRONMENT AND PARAMETER SET}\]

We use the ndnSIM platform to verify the performance of our PCSRC with CEE [16] and ProbCache [17]. The experimental network has 31 routing nodes which contain 10000 contents as shown in Figure 6. Each content is divided into 10 chunks and each chunk is 10 Mbytes. The content request can come from leaf nodes only. The cache capacity of each node is 1000Mbytes. We suppose that the probability of requests obeys probability of Zip [32] and the arrival requests obey the parameters \( \lambda \) of Poisson process, where \( \alpha = 0.8 \).

We set the simulation time, transmission interval, and \( T_{\text{basic}} \) to 100s, 100ms, and 10s [27], respectively. The sojourn time of temporary caching content chunks is set to two times of the
content transmission interval. The sliding window is set to 3s and the chunks are multiplied by the first content chunk. We first introduce some performance parameters as follows.

1) CACHE HIT RATIO
Cache Hit Ratio (CHR) is the ratio of the response number of requests and the total number of requests in specific time. The calculation formula is defined as

$$CHR = \frac{resp}{count},$$  \hspace{1cm} (7)

where resp represents the response number for the cached node, count represents the number of requests received by nodes. The value of CHR reflects the advantage of a caching scheme. The higher the CHR is, the greater the validity of a caching scheme is, and the less redundancy of content is.

2) AVERAGE REQUEST HOP
Average Request Hop (ARH) refers to the average number of routes transmitted when an interest packet hits the request content. The formula is defined as

$$ARH = \frac{\sum_{rep \in REP} g_{rep}}{\sum_{rep \in REP} Q_{rep}},$$  \hspace{1cm} (8)

where $g_{rep}$ denotes the number of hops that an interest packet is responded, $Fre$ represents the number of response that an interest packet is responded, $REP$ denotes the set of user’s request. The lower the value of ARH is, the less the hops of response are. This value reflects the impact of the cache mechanism on the bandwidth.

3) SERVER TRAFFIC RATIO
Server Traffic Ratio (STR) denotes the traffic ratio of the requests responded by CSS and the total requests, which is defined as

$$STR = \frac{\sum_{rep\in REP} W_{rep}}{\sum_{rep\in REP} Q_{rep}},$$  \hspace{1cm} (9)

FIGURE 5. The user $U_1$ requests the content chunk ($\Theta_{k,1}$).
where $W_{rep}$ shows the traffic which is generated by the request responded by CSS, $Q_{rep}$ represents the total traffic in the network. The higher the value of STR is, the lower the probability of requests responded in the cache path is. The higher the value of STR is, the bigger the value of hop of obtaining a content is, and the greater the overload is.

4) AVERAGE REQUEST DELAY
Average Request Delay (ARD) represents the average delay from sending interest packets to receiving response data packets.

B. SIMULATION RESULT
1) CACHE HIT RATIO
Figure 7 shows the CHR of PCSRC under the weight parameter $\gamma = \{1, 5, 8, 20\}$. From Figure 7, the CHR is the highest with $\gamma = 5$, while it reaches to the rock-bottom with $\gamma = 20$. However, when $\gamma = 1$ or $\gamma = 8$, the CHR is relatively reduced. When the weight parameter $\gamma$ is 20, it has the lowest CHR. When the weight parameter $\gamma$ are set as 1, 5, 8 and 20, the corresponding average CHRs are 38%, 43%, 40% and 34%, respectively. This situation is mainly due to the influence of the value of $\gamma$. The value of $\gamma$ not only takes account of the request in the history request window, but also takes account of the impact of the request in the current window. So PCSRC has higher cache hit rate. Of course, a reasonable parameter value has a great influence on improving the CHR.

![FIGURE 7. CHR of PCSRC by increasing cache space under different weight parameter $\gamma$.](image)

Figure 8 shows the CHR of PCSRC in comparison with CEE and ProbCache, where the cache space of each router is set to 5%, 10%, 15% and 20%, respectively. From Figure 8, the CHR of PCSRC is higher than those of CEE and ProbCache. The CHRs of PCSRC, ProbCache and CEE reach 42.7%, 25.3% and 21.4% under different cache spaces, respectively. The CHR of PCSRC is more 43% and 50% than those of ProbCache and CEE, respectively. The reason is that PCSRC performs pre-caching of content so that users can obtain the content nearby. The CHR of CEE is relatively low due to its blind caching strategy.

2) AVERAGE REQUEST HOP
Figure 9 shows the ARH in comparison with CEE and ProbCache by increasing cache capacity under $\gamma = 5, \alpha = 0.8$.

From Figure 9, ARH of each strategy gradually decreases with the increase of cache capacity, since the requested content is gradually cached in routers. The ARH of CEE is about 3.9 which is the largest in the three schemes due to its strategy of caching everything everywhere. Since ProbCache stores content in a path based on a fixed probability (setting 0.7 in our simulation) without taking the correlation of requests into account, its ARH is about 3.5, which is a little better than that of CEE. Compared to the two schemes, the ARH of PCSRC can reach about 2.4, since PCSRC actively caches subsequent contents according to the request content, and the gradually pushes the popular content to the edge of the network.

![FIGURE 8. The CHR by increasing the cache space.](image)

Figure 10 shows the ARH in comparison with CEE and ProbCache by increasing the parameter $\alpha$. From Figure 10, the ARH of each scheme is decreasing with the $\alpha$ increasing. The average values of ARH of PCSRC, ProbCache and CEE are 2.82, 4.01 and 4.17, repectively. Specifically, the ARH of PCSRC is 1.42 hops lower than that of ProbCache and 1.6 hops lower than that of CEE.

3) SERVER TRAFFIC RATIO
Figure 11 shows the STR in comparison with CEE and ProbCache by increasing the cache capacity. The STR of CEE is only 0.36 when the cache space is 20%, which does not improve significantly with increasing the cache capacity, since CEE makes the cached content to be homogeneous. The STR of ProbCache decreases to 0.26 when the cache
of each scheme. The ARD of CEE is the largest due to its caching everything everywhere. The ARD of ProbCache is a little worse than that of PCSRC since it caches the content on the path with a fixed probability. The ARD of PCSRC can get the best than those of CEE and ProbCache, because it pre-caches the subsequent chunks of current content.

V. CONCLUSION

As an important feature of IC-IoT, cache is an important means to improve the performance of smart devices in the network. In this paper, we focus on the caching strategy and proposed a Pre-Caching Strategy based on the Relevance of requested Content for Collaboration between nodes, called PCSRC. In PCSRC, the subsequent content chunks of a request are pre-cached based on the relevance of content. Furthermore, we set a sojourn time for each content chunk according to its popularity. On the one hand, PCSRC can push the popular content to the edge of the network efficiently. On the other, PCSRC reduces the hop count, which improves the user experience. We verify the validity of the method from four aspects: the first is cache hit ratio, and PCSRC can reach 42.7% with the different storage. This benefits largely from the pre-caching of content so that users can obtain the content nearby; the second is the average hop count, and PCSRC can reach about 2.4, making for average hop count as high as 47.9% than CEE; the third is the STR. Compared with the other two schemes the STR of PCSRC can reach 0.073 when the cache space is same; the fourth is ARD. The ARD of PCSRC can get the best than CEE and Prob Cache under different parameters $\lambda$. The above simulation results show that the PCSRC outperform existing methods, for example, CEE and Prob Cache. In the future, we will further enhance our schemes for large data and emulate our schemes in the different network topology.

REFERENCES

[1] G. Liu, W. Quan, N. Cheng, H. Zhang, and S. Yu, “Efficient DDoS attacks mitigation for stateful forwarding in Internet of Things,” J. Netw. Comput. Appl., vol. 130, pp. 1–13, Mar. 2019.

[2] Z.-Y. Ai, Y.-T. Zhou, and F. Song, “A smart collaborative routing protocol for reliable data diffusion in IoT scenarios,” Sensors, vol. 18, no. 6, p. 1926, 1926.
REFERENCES

[1] M. Zhang, M. Yang, Q. Wu, R. Zheng, and J. Zhu, “Smart perception and
descriptive optimization: A novel bio-inspired hybrid routing protocol for
MANETs,” *Future Gener. Comput. Syst.*, vol. 81, pp. 505–513, Apr. 2018.

[2] M. Zhang, B. Hao, F. Song, M. Yang, J. Zhu, and Q. Wu, “Smart collab-
orative video caching for energy efficiency in cognitive context centric
networks,” *J. Netw. Comput. Syst.*, vol. 158, May 2020, Art. no. 102587, 
doi: 10.1016/j.jncs.2020.102587.

[3] F. Song, Y.-T. Zhou, Y. Wang, T.-M. Zhao, I. You, and H.-K. Zhang,
“Smart collaborative distribution for privacy enhancement in moving target
defense,” *Inf. Sci.*, vol. 479, pp. 593–606, Apr. 2019.

[4] X. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen,
“Space/aerial-assisted computing offloading for IoT applications: A
learning-based approach,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5,
p. 1117–1129, May 2019.

[5] Q. Ge, C. Wen, and S. Duan, “Fire localization based on range-range
model for limited interior space,” *IEEE Trans. Instrum. Meas.*, vol. 63,
no. 9, pp. 2223–2237, Sep. 2014.

[6] Q. Ge, T. Shao, Q. Yang, X. Shen, and C. Wen, “Multisensor nonlinear
fusion methods based on adaptive ensemble fifth-degree iterated cubic
information filter for biomechatronics,” *IEEE Trans. Syst., Man, Cybern.
Syst.*, vol. 46, no. 7, pp. 912–925, Jul. 2016.

[7] K. Deng, L. Xing, L. Zheng, H. Wu, P. Xie, and F. Gao, “A user identifica-
tion algorithm based on user behavior analysis in social networks,” *IEEE
Access*, vol. 7, pp. 47114–47123, 2019.

[8] M. Zhang, Y. Zhou, W. Quan, J. Zhu, R. Zheng, and Q. Wu,
“Online learning for IoT optimization: A Frank-Wolfe Adam based
algorithm,” *IEEE Internet Things J.*, early access, Mar. 30, 2020, doi:
10.1109/JIOT.2020.2984011.

[9] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen,
“Airground integrated mobile edge networks: Architecture, challenges, and
opportunities,” *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26–32, Aug. 2018.

[10] H. Zhang, W. Quan, H.-C. Chao, and C. Qiao, “Smart identifier network:
A collaborative architecture for the future Internet,” *IEEE Netw.*, vol. 30,
no. 3, pp. 46–51, May 2016.

[11] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley,
C. Papadopoulos, L. Wang, and B. Zhang, “Named data networking,”
*ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.

[12] B. Feng, H. Zhang, H. Zhou, and S. Yu, “Locator/identifier split networking:
A promising future Internet architecture,” *IEEE Commun. Surveys Tutx.*,
vol. 19, no. 4, pp. 2927–2948, Jul. 2017.

[13] M. Zhang, P. Xie, J. Zhu, Q. Wu, R. Zheng, and H. Zhang, “NCPP-based
caching and NUR-based resource allocation for information-centric net-
working,” *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 5, pp. 1739–1745,
May 2019.

[14] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and
R. L. Braynard, “Networking named content,” *IEEE Commun. Mag.*,
vol. 44, no. 3, pp. 66–73, 2006.

[15] I. Psaras, W. K. Chai, and G. Pavlou, “Probabilistic in-network caching
for information-centric networks,” in *Proc. Inf.-Centric Netw. Workshop
(ICN)*, Helsinki, Finland, 2012, pp. 55–60.

[16] Y. Zhang, X. Tan, and W. Li, “PPC: Popularity prediction caching in ICN,”
*IEEE Commun. Lett.*, vol. 22, no. 1, pp. 5–8, Jan. 2018.

[17] G. Ge, Y. Guo, C. Liu, and J. Lan, “Collaborative caching algorithm
based on request correlation in named data networking,” *J. Electron. Inf.
Technol.*, vol. 36, no. 12, pp. 2795–2801, 2014.

[18] N. Laoutaris, S. Syntila, and I. Stavrakakis, “Meta algorithms for hierar-
chical Web caches,” in *Proc. 23rd IEEE Int. Perform. Comput. Commun.
Conf. (IPCCC)*, Phoenix, AZ, USA, Apr. 2004, pp. 445–452.

[19] Y. Chae, K. Guo, M. M. Buddhikot, S. Suri, and E. W. Zegura, “Silos,
rainbow, and caching token: Schemes for scalable, fault tolerant stream
caching,” *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1328–1344,
Sep. 2002.

[20] W. K. Chai, D. He, I. Psaras, and G. Pavlou, “Cache ‘less for more’ in
information-centric networks,” in *Proc. 11th Int. IFIP TC Net. (NET-
WORKING)*, Prague, Czech Republic, 2012, pp. 27–40.

[21] S. Wang, J. Bi, J. Wu, Z. Li, W. Zhang, and X. Yang, “Could in-network
caching benefit information-centric networking?” in *Proc. Asian Internet
Eng. Conf. (AINTEC)*, Bangkok, Thailand, 2011, pp. 112–115.

[22] G. Mauri, M. Gerla, F. Bruno, M. Cesana, and G. Vetrabile, “Optimal con-
tent prefetching in NDN vehicle-to-infrastructure scenario,” *IEEE Trans.
Veh. Technol.*, vol. 66, no. 3, pp. 2513–2525, Mar. 2017.

[23] M. Mangili, F. Martignon, S. Paris, and A. Capone, “Bandwidth and cache
leasing in wireless information centric networks: A game theoretic study,”
*IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 679–695, Jan. 2017.

[24] M. Zhang, Y. Yu, W. Guan, Z. Zhu, and J. Zhu, “WAVE: Popularity-
based and collaborative in-network caching for content-oriented networks,”
in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)* Workshops, Orlando,
FL, USA, Mar. 2012, pp. 316–321.

[25] Z. Ming, M. Xu, and D. Wang, “Age-based cooperative caching in
information-centric networking,” in *Proc. 23rd Int. Conf. Comput. Com-
mun. Netw. (ICCCN)*, Shanghai, China, Aug. 2014, pp. 1–8.

[26] 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 Conf. Comput. Commun. (INFOCOM)*
Workshops, Orlando, FL, USA, Mar. 2012, pp. 316–321.

[27] Z. Ming, M. Xu, and D. Wang, “Age-based cooperative caching in
information-centric networking,” in *Proc. IEEE Conf. Comput. Commun.
Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Apr. 2014, pp. 470–475.

[28] H. Wu, J. Li, and J. Zhu, “MBP: A max-benefit probability-based caching
strategy in information-centric networking,” in *Proc. IEEE Int. Conf.
Commun. (ICC)*, London, U.K., Jun. 2015, pp. 5646–5651.

[29] T. Liu, M. Zhang, J. Zhu, R. Zheng, R. Liu, and Q. Wu, “ACCP: Adaptive
congestion control protocol in named data networking based on deep
learning,” *Neural Comput. Appl.*, vol. 31, no. 9, pp. 4675–4683, Sep. 2019.

[30] M. F. Majeed, M. N. Dailey, R. Khan, and A. Tunpan, “Pre-caching: A
proactive scheme for caching video traffic in named data mesh net-
works,” *J. Netw. Comput. Appl.*, vol. 87, pp. 116–130, Jun. 2017.

[31] R. Chiorochetti, D. Rossi, G. Rossini, G. Carrofill, and D. Perino, “Exploit
the known or explore the unknown?: Hamlet-like doubts in ICN,” in *Proc.
Inf.-Centric Netw. (ICN)*, Helsinki, Finland, 2012, pp. 7–12.