IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2020.DOI

A Lightweight Caching Decision Strategy Based on Node Edge-Degree for Information Centric Networking

QIFENG YANG[12], HAOJIANG. DENG[12], AND LINGFANG WANG[12].
[1]National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences No. 21, North 4th Ring Road, Haidian District, Beijing 100190, P. R. China(e-mail: {yangqf, denghj, wanglf}@dsp.ac.cn)
[2]University of Chinese Academy of Sciences, Beijing 100190, P.R.China

Corresponding author: Haojiang Deng (e-mail: Denghj@dsp.ac.cn).

This work was supported in part by "Strategic Priority Research Program of Chinese Academy of Sciences: Standardization research and system development of SEANET technology" National Science and Technology of Major Projects (No.XDC02070100).

ABSTRACT In-network caching as a core function in ICN can greatly improve the content distribution efficiency and reduce redundant network traffic. It is a challenge to design caching strategies in ICN to improve the performance of in-network caching. Traditional lightweight strategies are inefficient due to problems such as edge caching pollution and slow diffusion speed. Current caching strategy with popularity prediction does not fully consider the content capacity of ICN network. They usually have huge computation overhead and cannot meet line-speed requirement. This paper focuses on the research of lightweight on-path caching. Our caching strategy is proposed according to the edge-degree of routers along the forwarding path. This strategy can quickly form a hierarchical content placement with little overhead. A large number of experiments are conducted to examine the performance of our strategy in different topologies and workloads. The simulation result reveals that our caching strategy can reduce the average download latency by 10.2% and increase the cache hit ratio by 9.6% at most compared with other lightweight strategies.

INDEX TERMS Information-Centric Networking, In-network caching, Caching strategy, Lightweight

I. INTRODUCTION

Information-Centric Networking (ICN) [1] is proposed as an effective future Internet architecture to address the problem of content distribution efficiency in current network architecture. Pervasive in-network caching is regarded as a fundamental function of ICN [2]. The pervasiveness makes ICN's caching mechanism different from traditional Web caching. The caching network topology in ICN is no longer hierarchical but arbitrary [3]. Meanwhile, as basic service, caching in ICN router should support line-speed operation [4], which limits the computation consumption of each router and the caching strategies must be simple enough [5]. Caching strategies determine the content placement in the network [6]. The content placement is a process in which the user's requested content is cached in the caching nodes along the forwarding path. It has a significant role to improve the overall network performance [7] [8].

ICN caching strategies can be divided into three types: individual caching, on-path caching and collaborative caching. Simple as it is, individual caching usually have poor performance due to the limited information in each node. In on-path caching, a content is cached in the routers along the return path when it is served from the source of the content to the user, so the information along the path can be used to improve the performance of strategies. Collaborative caching strategies often need to communicate with the nodes beyond the forwarding path but they introduce too much additional overhead to the device and bandwidth to support the line-speed operation. For example, some collaborative off-path caching strategies add too many mechanisms to the network, resulting in heavy load for the link and decrease in node performance [9] [10]. Therefore, these strategies are out of our scope and this article will only focus on the on-path caching strategies.

The primary goal of on-path caching strategies is to cache the replica of popular content to the edge of the network as quick as possible [11]. Traditional on-path caching strategies may cause unpopular contents to occupy the cache due to the simple mechanism, or the popular contents can not spread to the edge of the network quickly [12]. Some recent strategies make caching decisions in a priori way by counting the request times of passing content, which can significantly improve the hit ratio of cache compared to the traditional strategies [13] [14]. However, these strategies tend to conservatively estimate the huge content space constraint. It is difficult for hardware devices to implement these idealized strategies due to the cache capacity limitation. Therefore, we design a lightweight caching strategy that can handle network traffic at line-speed, using filtering mechanism to prevent content that is only requested once from being cached to the edge of the network. We collect route information instead of popularity information to spread the replica of popular content quickly to the edge of the network.

The main contributions of this paper are as follows.

1) We formulate the caching revenue maximization problem as the average latency minimization problem in the general cache system model and propose the ideal content placement in the model. Different caching strategies are compared and evaluated according to this model.
2) We propose a Node Edge-degree-based Caching strategy to push popular content to the edge of the network quickly and to help forming a hierarchical content placement in the network. We implement our strategy in our self-developed content router based on Protocol-Oblivious Forwarding(POF) switches and design corresponding protocols and flow tables.
3) We evaluate our caching strategy in simulations based on real-world network topologies and various workloads, which reveals that our strategy can reduce the average download latency and increase the cache hit ratio. Throughput and computational overhead of our strategy implemented in content router are tested to verify the scalability of our design.

The rest of this paper as follows: Section II provides related research work on ICN on-path caching strategies. System model is presented as Section III. Section IV presents the Node Edge-degree-based Caching Strategy. Section V is experimental results. Section VI concludes the paper.

II. RELATED WORK

On-path caching is performed at the routers of a network, following either a chunk or a packet naming granularity. On-path caching is highly influenced by the dynamic networking enviroment of ICN, which means operations such as monitoring and collection of statistical information tend to be impractical and inefficient. Due to these limitations, on-path caching is considered to be a short-term decision where the lighweight and heuristic tecnhiques are usually used [15].

Many current on-path strategies filter the content based on popularity [20]. Only the most popular content can be cached in the edge routers while unpopular content may only be cached in the center of the network. MPC [13] records the popularity of content items by adding a counter for each entry in the hash table and predicts the popularity of content based on the number of requests for the content in the past. However, it needs to monitor all the content requests passingby, which consumes a large amount of CPU resources and dram memory. The overhead is so large that line-speed requirement can not be met and the current cache capacity is not enough to hold all the record. Despite the trade-off between popularity accuracy and memory consumption in a multi Bloom filter strategy [18] and CBF-Hash tables hybrid scheme [19], the forwarding performance of the routers are severely affected unless consuming huge amounts of memory [20]. Other proactive strategies try to cache content only in the wireless edge [21] [22], they are just optimization of the edge cache, ignoring consideration of the overall network caching performance.

Strategies mentioned above act in a priori way. They predict the content popularity as prior knowledge and make caching decision according to it with huge memory consumption. Some traditional lightweight strategies make caching decisions without prior knowledge and cache content frequently with simple replacement strategies used [23]. Gradually, content that remains in the cache is supposed to be popular. These strategies use meta algorithms to determine the caching location of a content item with little overhead. LCE as the default algorithm caches the content in all the nodes along the path by default, which leads to much redundancy. EDGE caching [24] [25] caches all the content items at the edge of network. However, this strategy will run out of edge cache capacity and result in fast content replacement. Replacement errors will happen frequently due to the existence of a very high percentage of objects that are requested only once. These so called one-timer objects usually amount up to 45% of the total requests and 75% of the total distinct objects presented in the workload [26]. The caching for these one-timer objects will result in reduced performance. ProbCache [11] uses a probabilistic approach to cache content in a subset of on-path nodes, which reduces the frequency of edge replacement. But the proportion of one-timer objects on the edge is still not changed. Both popular and unpopular content are cached in the edge without distinction, which we call edge caching pollution.

In LCD [12], a new copy of the requested content item is cached only at one hop downstream on the path from the location of the hit router to the user. LCD is more “conservative” than LCE or ProbCache as it requires multiple requests to bring a content item to an edge router, with each request advancing a new copy of the content item one hop closer to the user. So one-timer content items are hardly cached in the edge and rate of replacement errors will be greatly reduced. But for the same reason, the spread of popular content is also suppressed because a popular content item can not be cached in the edge before being requested for enough times. The slow diffusion speed brings a bad time adaptability. There will also be much redundancy along the path, which will affect the performance of caching.

Other on-path strategies in [27] [28] are based on global topological information of nodes. These strategies believe that content should be cached at a node with higher centrality.

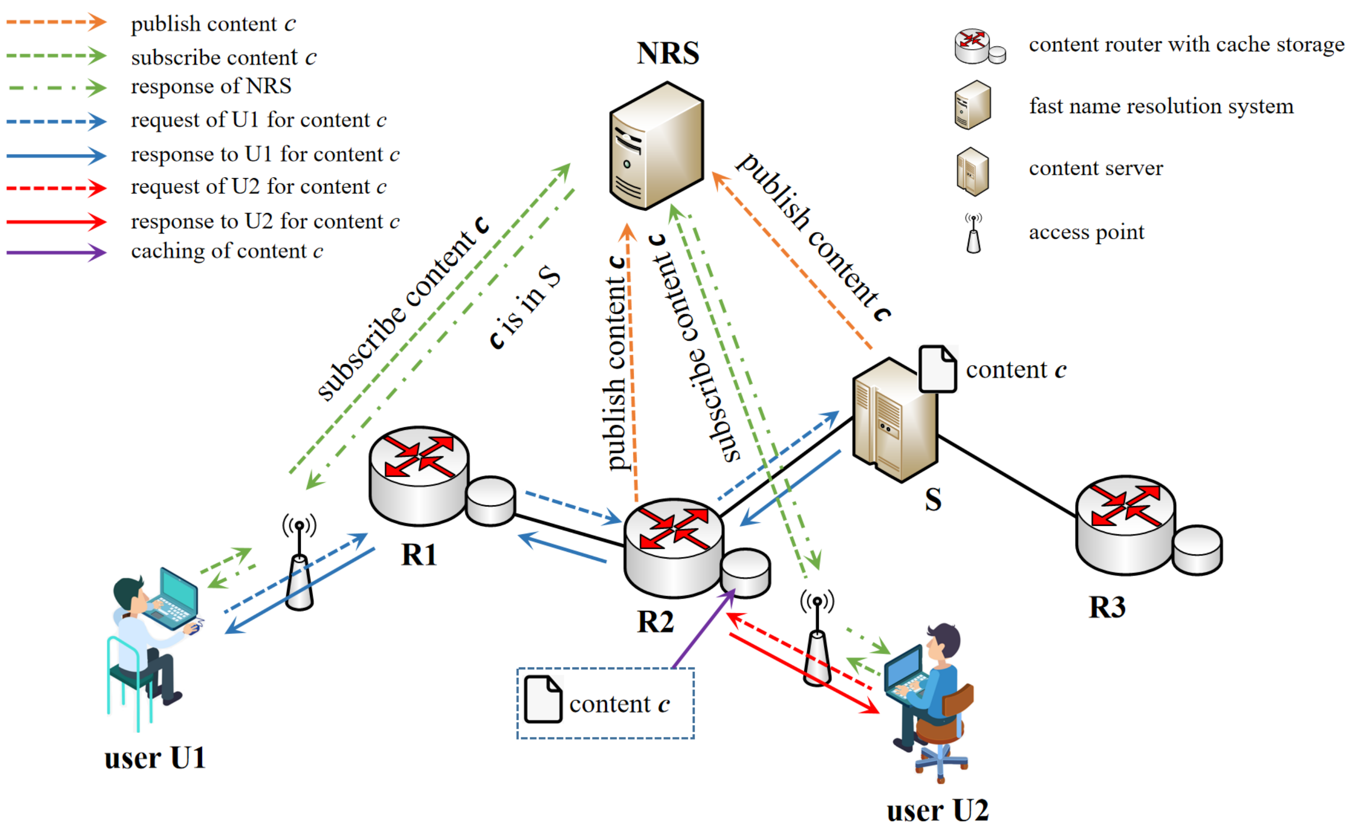


FIGURE 1. A generic caching mechanism in an ICN network.

Many centrality measures are proposed including Degree Centrality, Stress Centrality, Betweenness Centrality, Closeness Centrality, Graph Centrality and Eccentricity Centrality. Except the Degree Centrality, the other centrality measures are impossible to be calculated locally. The system has to have a holistic view of the topology and manage the dynamic topology with very high cost. Moreover, these strategies can improve the cache hit ratio by putting popular content copies in the center of networks, but they cannot effectively reduce the download latency because requests have to travel multiple routers to hit the node with high centrality.

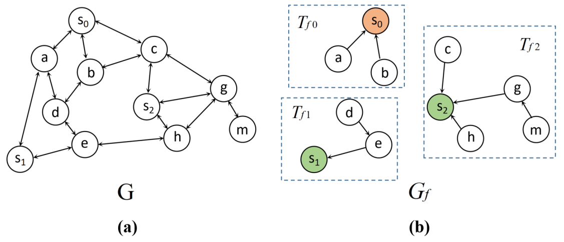


FIGURE 2. Network topology and the genration process of forwarding path for f. Nodes in forwarding tree T_{fi} will forward local request for content f to the node s_i along the shortest path. All these forwarding tree T_{fi} make up forwarding forest G_f.

III. SYSTEM MODEL

A. MODEL SET UP

In this section, we model and study a general caching network in ICN. As shown in Fig.1, before requesting a content c in the caching network, user U1 needs to ask the local fast name resolution system(NRS) where is the nearest content replica. Then U1 will send an interest packet(request) to the corresponding node S(which is the server of content c and published content c in NRS previously) when receiving the response from the NRS. Once the request hits the node, content will be retrieved along the symmetrical path S-R3-R2. All routers in the network are equiped with caching abilities and R3 decides to cache the content c according to the local caching strategy and publishes it to the NRS. When user U2 requests the content c, the NRS will respond to U2 with the node R3. So U2 can quickly get the content c at node R3 without travelling to S.

A possible topology of network is an arbitrary directed graph G= $\langle V, E\rangle$ as shown in Fig.2, where node set V represents content routers and edge set E is the set of links between these nodes. C_i is the cache size of node $i \in V$ and $(i, j) \in E$ denotes the link between node i and node j.

The request generated by users attached to each node is i.i.d. and follows a Poisson process based on the parameter λ. The content set is $F = \{f_1, f_2, ...\}$ with a size of $|F|$. Contents in the network exist in chunks. Each chunk corresponds to a content request. Thus, each chunk can be considered as an item. We commonly set each content item following Zip-f distribution with a unified size M [16]. A content server s_0 is set in the network with infinite storage to hold all the content items in F for which all the request can be served in the network. For the content item f requested by any user, the total number of content copies in the network is denoted by $k(k >= 0)$, and the set of nodes storing the content item f is $S_f = \{s_0, s_1, s_2, ...s_k\}$.

TABLE 1. List of main notations used in the system model

notation	representation
V	node Set
E	edge Set
(i, j)	links between node i and node j
C_i	cache size of node i
F	content set
f_i	i^{th} content in F
S_f	set of nodes which have content f
s_0	server node which has long-term storage for all content in F
$s_i(i! = 0)$	an arbitrary node in S_f caching content f
D_{ij}	latency of transmitting a content item along (i, j)
T_{fi}	forwarding tree for content f which is rooted at s_i
G_f	forwarding forest which is composed by T_{fi} for all $s_i \in S_f$
r_i^f	request rate for content f by direct users in node i
R_{ij}^f	request rate for content f in node j from node i
X_i^f	boolean variable indicates whether content f is in node i

When a user requests a content, the content resolution system will automatically respond the user with the closest node among all the node caching this content item. Then the user will send a request message to that node. Here we assume that the request message will be sent along the path with the minimum distance. The latency of transmitting a content item between any two neighbour nodes i,j is the sum of transmission latency and processing latency. We set the latency as a constant value D_{ij} and use the latency D_{ij} to measure the distance between i and j. The network devices and link bandwidth can process and forward all the traffic in time without denial of service or network congestion.

B. PROBLEM FORMULATION AND ANALYSIS

Average download latency is an important indicator to measure performance of network caching, which is the time interval between when the content request is sent and the data packet arrives. In this subsection, we will formalize the average download latency minimization problem, then analyse the impact of related parameters and factors.

Given the content distribution status in the network, for any request for content item f, we first construct a path tree to determine the node where users can request content item f. The set of sources for content item f is S_f= $\{s_0, s_1, s_2, ...s_k\}$ as shown in Fig.2(where k = 2). The nearest source s_i will be chosen for each node to forward the content request and the shortest path can be determined. For example, node s_1 will be chosen as the source of content item f for node e because it is closest to node e among $\{s_0, s_1, s_2\}$. At each node there must be one and only one next hop to forward an unsatisfied request for content f. Therefore, a tree (T_{fi} = $\langle V_{fi}, E_{fi}\rangle$) rooted at s_i can be derived from these forwarding path (Fig.2(b)), where V_{fi} is a subset of V and E_{fi} is a subset of E. As shown in Fig.2(b), requests for content item f in node c will only be forwarded to s_1, so the edge between s_0 and c is neither included in T_{f1} or T_{f0}. Then we can get the forwarding path map G_f composed by all the $k + 1$ path trees as shown in Fig.2(b), where G_f = $T_{f0} \cup T_{f1} \cup ..T_{fk}$. An unsatisfied request for f at node i will be forwarded to the parent node of i in G_f, from which we can derive forwarding path of all the requests for content item f at any node.

Now that we have the request path of each node for the content item f according to the spanning forest G_f, which can be derived from all the connected graph T_{fi}, we can start to calculate the download latency. Here we define r_i^f as the average local request arrival rate for content f in node i, which is contributed by local users. And we define R_{ij}^f as the request rate from node i to node j for f. Therefore, in node i, the request rate for content f is r_i^f plus the sum of the request rate of its child nodes in G_f, the request rate will be ended at the node if it stores the content item f, otherwise the request will be forwarded towards the parent node as illustrated in Fig.3. So we have R_{ij}^f for $(i, j) \in G_f$:

$$R_{ij}^f = (r_i^f + \sum_{k:(k,i)\in G_f} R_{ki}^f)(1 - X_i^f) \tag{1}$$

X_i^f is 1 if content f is in node i and 0 otherwise. Because D_{ij} represents the physical transmission latency between nodes i, j. According to our assumption that all packets travel in a shortest path, each content request forwarded upstream traversing link (i, j) corresponds to a content chunk traversing the reverse link (j, i) downstream. The requests of each node obey IRM distribution [33], and the total number of requests in the network per unit time is a steady-state value which can be set as a constant Q. So we get the average download latency(ADL) function:

$$ADL = \sum_{f\in F} \sum_{(i,j)\in G_f} 2R_{ij}^f D_{ji}/Q \tag{2}$$

The average download latency minimization problem can be formulated as follows:

$$min \sum_{f\in F} \sum_{(i,j)\in G_f} 2R_{ij}^f D_{ji}/Q \tag{3}$$

$$s.t.\ C_i \geq \sum_{f=1}^{F} X_i^f \qquad \forall i \in N, \forall k \in F \tag{4}$$

$$X_i^f \in \{0, 1\} \qquad \forall i \in N, \forall k \in F \tag{5}$$

$$X_i^f = 1 \qquad \forall i \in S, \forall k \in F \tag{6}$$

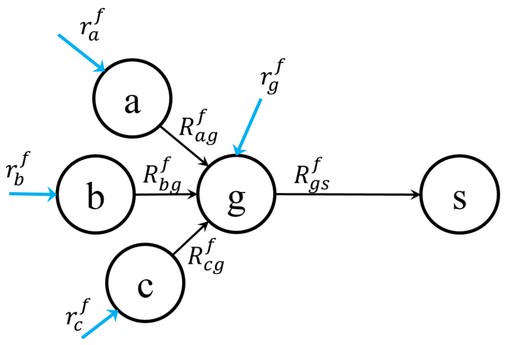


FIGURE 3. S is the source of content f and a, b, c, g do not have the content item f in cache. a, b, c are leaf nodes and R_{ag}^f is equal to r_a^f. g is parent of a, b and c, so R_{gs}^f is the sum of R_{ag}^f, R_{bg}^f, R_{cg}^f and r_g^f.

The construction of forwarding path map is an MDS problem [17]. The minimum average latency solution is essentially a static content placement problem which is a complete knapsack problem. Therefore, the problem can not be solved in polynomial time in a resource-constrained ICN router. When the network and content scale increase, the solution time will increase exponentially and line-speed requirement of ICN can not be meet.

Since it is a NPC problem unable to be solved in polynomial time, we analyse the formulation and can figure out from our experience that:

Proposition 1 Contents with higher popularity can reduce more latency.

We denote request rate from node i to node j for f as follows:

$$R_{ij}^f = (r_i^f + \sum_{k:(k,i)\in G_f} R_{ki}^f)(1 - X_i^f) = P_i^f(1 - X_i^f) \quad (7)$$

It means the traffic generated by each node is related to the content popularity P_i^f. if we have $X_i^{f_1} = X_i^{f_2} = 0$ and $P_i^{f_1} > P_i^{f_2}$, it will reduce more latency to cache f_1 than f_2 because target function has more reduction according to $P_i^{f_1}$ than $P_i^{f_2}$

Proposition 2 Along the path from content source to the user, caching farther away from source reduces more latency.

We consider two nodes $i, j \in G_f$, where i and j are on the path of the leaf node l to source node s. j is the parent node of i in G_f, and therefore closer to node s. For content f, $X_i^f = X_j^f = 0$, when storing at node j and not at node i, the total latency can be cut down by $D_j = \sum_{(m,n):(m,n)\in P_{js}} R_{mn}^f D_{nm}$ where P_{js} is the link set composed by the link from j to s in G_f, and when f is stored in i instead of j, the average latency will reduce $D_i = \sum_{(m,n):(m,n)\in P_{is}} R_{mn}^f D_{nm}$. We have $Di - Dj = R_{ij}^f D_{ji} > 0$, so caching farther away from source will cut down more downloading latency.

With Proposition 1 and 2, we conclude that content should be cached hierarchically in the network, the more popular content should be closer to the edge of the network.

IV. DESIGN OF NEC

In this section, the brief design idea about our strategy is proposed. We first describe the edge-degree estimations process in the algorithm, which is the foundation of caching decision in the strategy. Then we will show our Node Edge-degree-based Caching overall as well as the details and an example of the strategy.

A. EDGE-DEGREE ESTIMATION

According to the conculsion in Section II, the motivation of our research is to cache content hierarchically and quickly in the network. So the more popular content should be closer to the edge of the network and popular content needs as few request times as possible to diffuse to the edge . Explicit knowledge about popularity of content is not involved in our strategy for the lightweight consideration. We make caching decisions based on the node location information which implicitly reflects the popularity of content inside. Here we define the edge-degree as a basic parameter which measures how many hops a node is away from the nearest edge node. In traditional strategies, node information has to be maintained by central system, for example using SDN, because the node information only makes sense in the whole network. It will put heavy overhead on the controller when the network scale becomes large. So we will get the edge-degree in a distributed manner.

To get the edge-degree of all caching nodes in the network, the edge nodes are marked in advance with the edge-degree is zero by default. An edge node will periodically send broadcast packets to inform its one-hop neighbour of its edge-degree. After receiving the broadcast packet, the neighbour node will use the the edge-degree value plus one in the broadcast packet as a candidate value. The original value of edge-degree will be updated if it is larger than the candidate value. For example, as shown in Fig,4, the broadcast packet sent by node c to f tells f that the edge-degree of c is 0, and the default edge-degree of f is 16(16 is the default edge-degree for all caching nodes). So the edge-degree of node f is updated to 1. When f receives the broadcast packet from g, it will not update because the original edge-degree is 1, and the candidate edge-degree in the broadcast packet from g is 2. If the edge-degree of a node changes, it will send broadcast packets to inform its one-hop neighbour in the same way like an edge node, otherwise no operation is performed. The result of the update process of all nodes is shown in Fig.4.

B. NODE EDGE-DEGREE-BASED CACHING

In our target content placement, content in routers closer to the edge is supposed to be more popular. So a content hit in a router should be cached in a router with a smaller edge-degree. In our caching strategy, each time a content is hit or served, it is tend to be cached near the midpoint of the forwarding path. For a k-hop path, it takes $O(log(k))$ requests to bring the content to the edge of the network. It can be seen as a trade-off between Edge caching and LCD caching. It obviously reduces redundancy and avoids

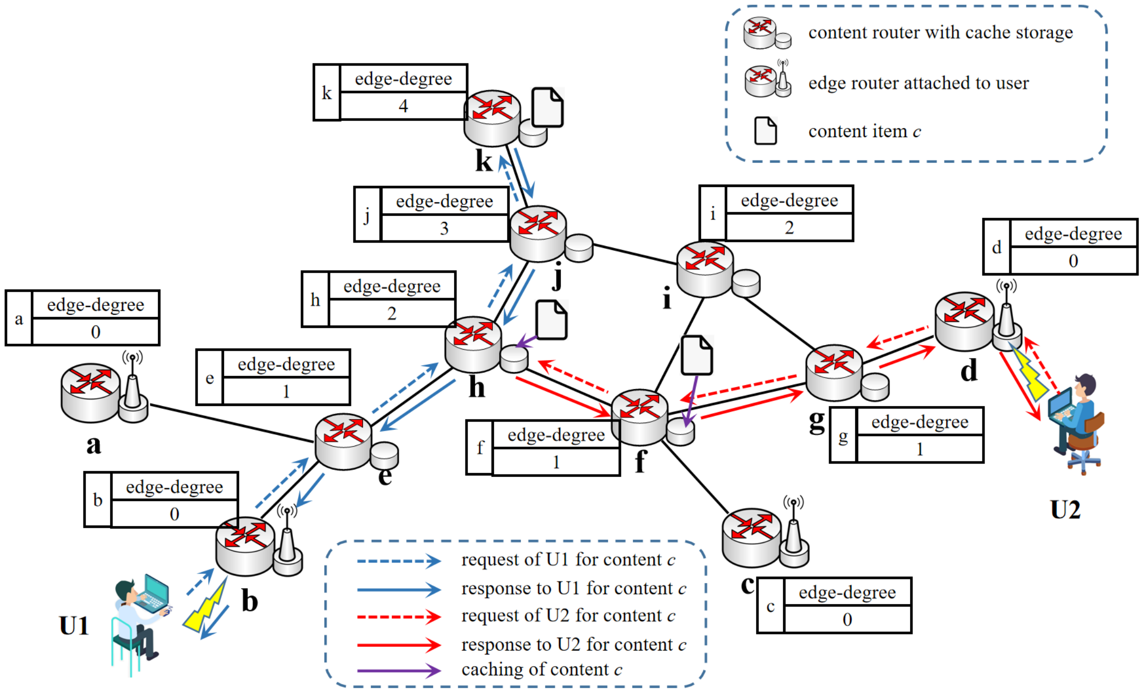


FIGURE 4. Caculation of edge-degree for caching node.

caching pollution in edge caching. Our caching strategy also accelerates content diffusion in LCD because LCD needs at least $O(k)$ requests to bring a popular content to the edge of the network compared with $O(log(k))$ requests in our strategy.

We use the edge-degree as a measure to find the midpoint of this path. For the node where a request is hit, if the edge-degree of the node is e, we select the node along the path with edge-degree $\lfloor e/2 \rfloor$ as the caching node. As shown in Fig.4, the request from U1 for content c is served at k and content c returns along $k-j-h-e-b$, the caching will take place at a node with an edge-degree of 2, which is node h, just near the midpoint of the forwarding path.

C. PROBLEM AND DISCUSSION IN NEC

Broadcast Storm. The process of calculating edge-degree requires a lot of interactions between neighbour nodes which may cause broadcast storm problems. Considering the worst case scenario, each node may cause n updates where n is in-degree of the node. The number of updates may be significantly large when recursing to the central node. Therefore, in each caching node, broadcast packets are sent only when the edge-degree updates, and the broadcast can not be sent before waiting for a delay (set to 100ms) to collect all the broadcast messages from neighours. Then the node will send the broadcast packet with local minimum edge-degree, which can effectively reduce the number of broadcast packets and the load of the central links and nodes.

Cache redundancy. Nodes on one given path may have the same edge-degree, which will cause duplicate caching of content on the path. Referring to Fig.4, when U2 request the content c from node h, data packets will be forwarded along the path of $h-f-g-d$. Node f and node g have the same edge-degree(1), which is exactly 1/2 of the edge-degree of node h(2). According to our caching mechanism, both f and g need to cache the content. To avoid redundancy, we will set a caching flag bit 1 in the content packet. The flag bit is set to 0 after caching. When the packet arrives at node g, no caching is performed because the flag bit has been set to 0 at node f.

Uneven Content distribution. The caching position of the content item sent from a certain node can be determined according to NEC. But the network topology is usually fixed or has little change, so the caching position from the same source along the same path is usually fixed. For example, when the content sent by node k travels through the path $k-j-i-g-d$, content can only be cached in i or downstreams. This will cause load imbalance and low cache space utilization in node j. Therefore, we need to distribute the caching from source to the midpoint of the path. So we adjust the value of $\lfloor e/2 \rfloor$ to $\lfloor e/2+p \rfloor$, where p is the discrete factor generated based on an uniform random generator to ensure a balanced caching distribution within the network.

Implementation. We implement our caching strategy in our POF-based ICN content router. The design of our soft router adopts a split architecture composed by a switch end and a storage end to prevent caching from blocking forwarding operations as shown in Fig.5 [29].The switch end is only

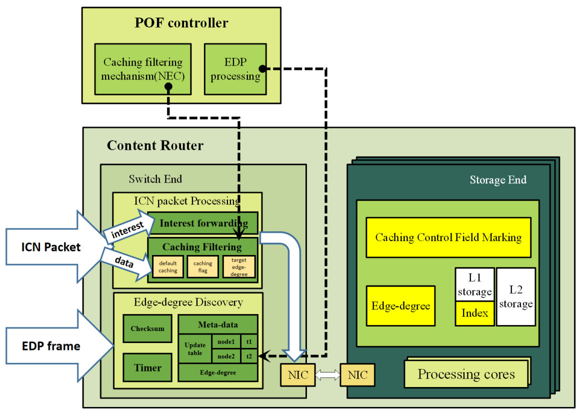


FIGURE 5. Architecture of POF-based content router.

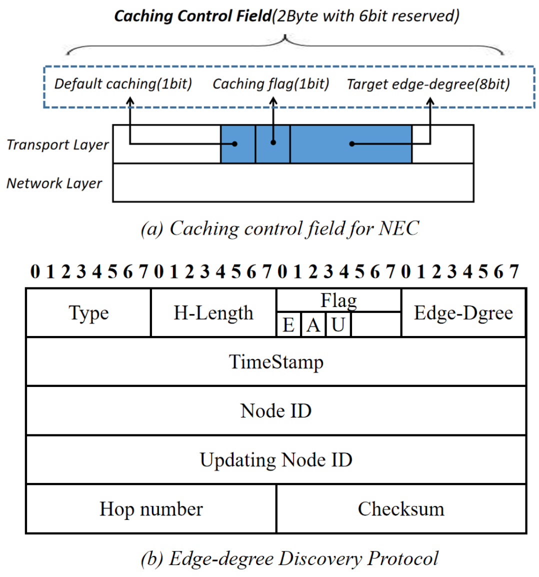


FIGURE 6. Packet header of NEC and protocol for EDP.

responsible for the filtering and forwarding of packets. We use a POF-switch to build the switch end enabling a white-box design so that flexible protocals and forwarding mechanisms can be supported [30]. The storage end manages the local storage with a DRAM-SSD-based hierarchical content store and communicates with the switch end through a NIC by our proprietary protocol.

When the content router receives an interest packet, the interest packet will be forwarded to the storage end. If the interest packet hits the content store in the storage end, the corresponding data packets for the interest will be encapsulated with caching control field in the packet header by the storage end. The caching control field for NEC is designed in header of transport layer and is shown in Fig.6(a). The caching control field marking process is shown in Algorithm 1. As the ICN standard is being developed [31], it cannot guarantee that all servers and producers support this strategy natively. In this case, an additional default caching field needs to be added to the caching control field, which is initially set to 1. All caching nodes will immediately cache the data packets marked by 1 in the default caching field and set the field to 0 after caching. So our router which supports the NEC strategy will set the default caching to 0 in storage end as line 4 in Algorithm 1. In the line 6 of Algorithm 1, caching flag is set to 1 according to previous description about cache redundancy in section IV.C. Target edge-degree is set to $\lfloor e/2 + p \rfloor$ where e is local edge-degree informed by the switch end and p is a discrete factor generated by Caching-Control-Field-Marking component ranged from 0 to $e/2$ in line 5. When the caching control field is marked into the header of the data packet, the packet is forwarded to the switch end and the switch end will forward the packet back towards the user.

When the data packet arrives at the router on the delivery path, the switch end of the router processes the packet first. Switch end will make a caching decision quickly according to the caching control field in the packet header. The decision making is shown as the data packet filtering in the algorithm 2. The default caching field in the header is first checked in line 4. If the default caching field is 1, the packet will be duplicated and forwarded to the storage end and next router respectively with default caching field reset to zero as shown in line 5. Otherwise the caching flag field is checked in line 7. If the caching flag is 0, the packet will not be cached. Otherwise, the following target edge-degree filed in the packet will be checked. Only when the target edge-degree in the packet is equal to the edge-degree in local meta-data, the caching can happen and the packet will be duplicated and forwarded to the storage as shown from line 8 to line 10.

Algorithm 1 Caching Control Field Marking

```
1:  Storage end gets a interest packet for $content$
2:  if $content \in$ local CS then
3:      read the local edge-degree from metadata as $e$
4:      $default\ caching = 0$
5:      discrete factor $p$ = randomint(0, $e/2$)
6:      $caching\ flag = 1$
7:      $target\ edge\ dgree = p + e$
8:      encapsulate the data packet with $default\ caching$,
        $caching\ flag$ and $target\ edge\ dgree$
9:      forward the data packet
10: else
11:     pass
12: end if
```

Edge-degree calculating is based on Edge-degree Discovery Protocol(EDP). Edge-degree discovery protocol is a point-to-point communication protocol used to exchange edge-degree information. It is neither an interest packet or a data packet. An EDP frame adopts Ethernet encapsulation format which should be friendly to the forwarding layer and easy to decode.

Algorithm 2 Data Packet Filtering

```
1: Switch end receives a data packet
2: read the local edge-degree as $e$
3: Get $caching\ flag$ field and $target\ edge\ degree$ field in
   the packet header
4: if $default\ caching == 1$ then
5:     duplicate the data packet to storage end
6:     $default\ caching \mathrel{\&}= 1$
7: else
8:     if $caching\ flag == 1$ then
9:         if $target\ edge\ degree == e$ then
10:            $caching\ flag \mathrel{\&}= 1$
11:            duplicate the data packet to storage end
12:        end if
13:    end if
14: end if
15: forward the data packet
```

In our design, we operate the EDP frames through POF controller. The protocal format is shown in Fig.6(b). Type and H-Length field are used to fast decode the frame. The Flag field includes error flag, ack flag and updating flag, which is used to mark different EDP frame types. The Node ID field marks the node which sends out the broadcast packet. The Updating Node ID field marks the node which causes the update of the node marked by Node ID. The Edge-degree field is the edge-degree of the node marked by Node ID. Timestamp is the time interval since last update in the node marked by Node ID caused by the Updating Node ID. While hop number is used for extension of multihop neighbour discovery.

When a content router j receives an edge-degree discovery frame from content router i, it first calculates the checksum. Then the TimeStamp and Updating Node ID are checked according to local update tables which records the time of the last update from all the neighbour nodes respectively. If Updating Node ID is local node ID and the time interval does not exceed the threshold, the frame will be discarded to avoid frequent updates. Otherwise, the edge-degree field will be compared with local meta-data. If the edge-degree field is smaller, local edge-degree and the update tables in in the meta-data will both be updated. Then the timer will be checked. If the time is up, new edge-degree discovery frames from content router j will be encapsulated. In the new encapsulated EDP frame, Node ID is Node ID of the content router j and Updating Node ID is Node ID of the content router i. The TimeStamp is 0 if the edge-degree of content router j is never updated by the content router i before. Otherwise, the TimeStamp is set to the time interval from now since last update. After encapsulation, the EDP frame will be broadcast from all ports of router j.

V. PERFORMANCE EVALUATION

TABLE 2. Simulation Environment

Prameter	value
Simulator	Icarus
Traffic Workload	Stationary/Youtube Vedio traffic
Catalog Size	$1 \times 10^6/581,527$
Topology	Cascade, Tree, GEANT, WIDE, TISCALI, GARR
Cache Size	Uniform
Replacement	LRU
Request number	$2\times10^6/1.46\times10^6$
Request distribution	Poisson
Request rate	10req/s

A. SIMULATION SETUP

We implement NEC in the Icarus simulator [32], which is a Python-based discrete-event simulator for evaluating caching performance in Information-Centric Networking(ICN). Icarus is not bound to any specific ICN architecture. Its design allows users to implement and evalute new caching policies or caching and routing strategies conveniently. We compare the performance of NEC against the most widely used strategy LCE, two of the best on-path caching strategies Leave Copy Down(LCD) [12] and Prob-Cache [11], and degree centrality based strategy CL4M [27]. We use LRU as a default replacement policy for all strategies. Caches are installed on all routers in the network and cache indexing is performed at the content level. Table 2 shows the basic parameters which are selected for our simulation. Cascade and Binary tree topology are used for testing basic characteristics of NEC. A total of four real world topologies are further used to evaluate performance of caching strategies. They are ISP-like topologies{nodes(consumer, caching node, source), edges}: GEANT{53(8, 32, 13), 61}, TISCALI{240(36, 160, 44), 810}, WIDE{30(6, 13, 11), 33} and GARR{61(21, 27, 13), 75}. Each result is averaged over ten distinct runs of the same scenario. Both stationary(IRM) [33] and real world traffic workloads are tested. The stationary workload uses a catalog of N = 1×10^6 content objects with popularity defined by the Zipf distribution (α = 0.8). A total of 2×10^6 requests are generated in these scenarios. The first 1×10^6 requests are used to allow caches to converge and are not used for gathering statistics; the remaining 1×10^6 requests are logged and used to gather statistics. In the real world temporal scenario, we use Youtube Vedio traffic generated by campus [34]. This workload includes a total of $581,527$ ditinct content items and 1.46×10^6 requests. Temporal locality varies between different content. We use the first 4.6×10^5 to warm up and the remaining to gather statistics.

B. EFFECT OF NEC PROPERTIES

Caching location. NEC expects content can be cached in midpoint of the delivery path. In order to observe the caching location in NEC strategy, we use a binary tree topology with depth of 8 for experiments. We record the length of path from each user to the serving node as T-length and the length

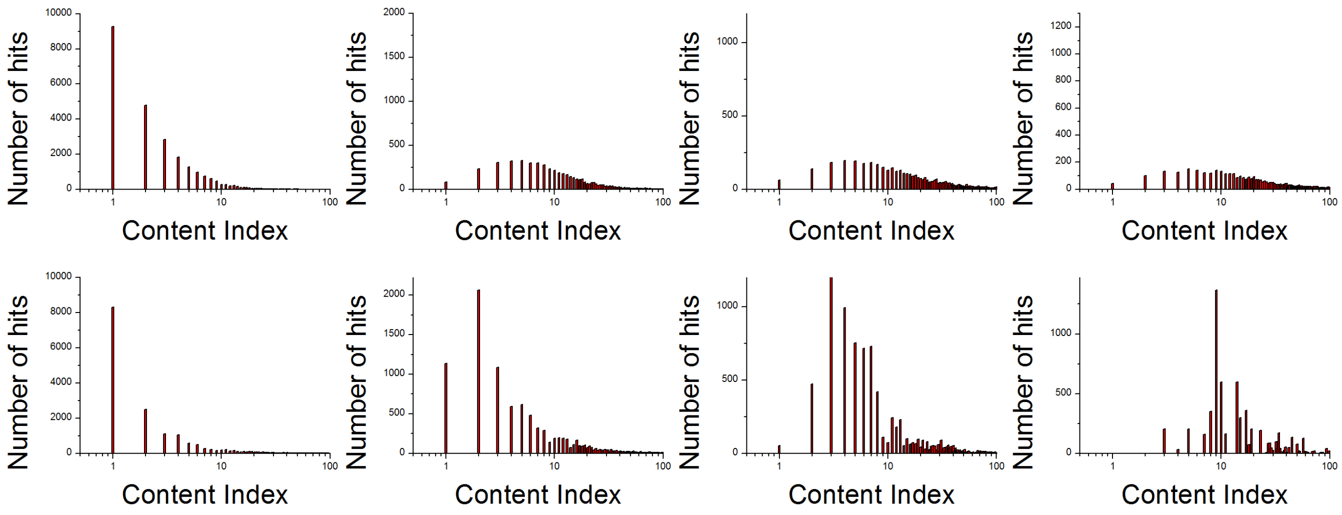


FIGURE 7. Content Hit times in nodes with different edge-degree.

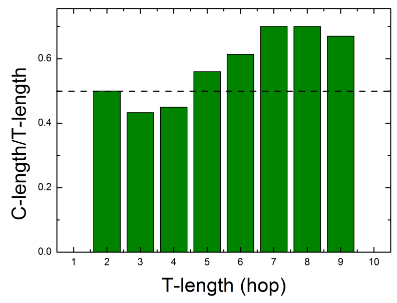


FIGURE 8. ratio of caching distance to total distance.

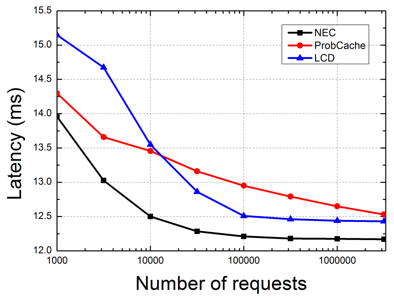


FIGURE 10. latency under different numbers of requests.

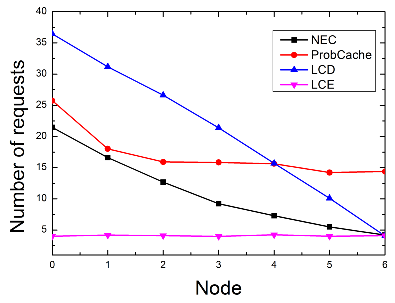


FIGURE 9. Expected times of request to cache content into node with different edge-degree.

of path from the user to the caching node as C-length. The length of path is denoted by number of hops. We take an average of C-length under the same T-length, and the ratio of C-length to T-length is shown in Fig.8. When the T-length is short, the ratio tends to be less than 1/2 which means caching location is closer to the user. This is because the serving node is close to the edge(requester) and the rounding error will cause the length to be shorter. Moreover, when the serving node is near the edge but not near the user, it will also cause the actual caching location far away from the midpoint. But in this case, because the content in the serving node can be considered as popular content, it should be cached closer to the edge. When T-length is large, the C-length is usually more than half of T-length, because the rounding has little impact on the overall, and the probability of the edge serving can be ignored. The discrete factor mentioned in Section III.C will make the distance from cache location to the user longer than the serving node.

Diffusion rate. NEC usually caches content in the midpoint of the path, so the diffusion of popular content to the edge is usually faster than LCD and slower than ProbCache.We tested the diffusion speed of the three strategies in an eight-layer cascaded linear topology. The experimental results are shown in the Fig.9. Each point (n, r) represents the average number of requests r required to diffuse content i (here we choose i as the most popular content to enhance the experimental effect) in node n or nodes closer to the user than node n. Obviously NEC outperforms LCD in terms of diffusion speed. Unexpectedly, NEC has a faster diffusion speed than ProbCache, which means NEC requires fewer requests to diffuse content to the edge. This may be due to the setting of caching probability in ProbCache. It is found that the diffusion speed of ProbCache is basically stable at

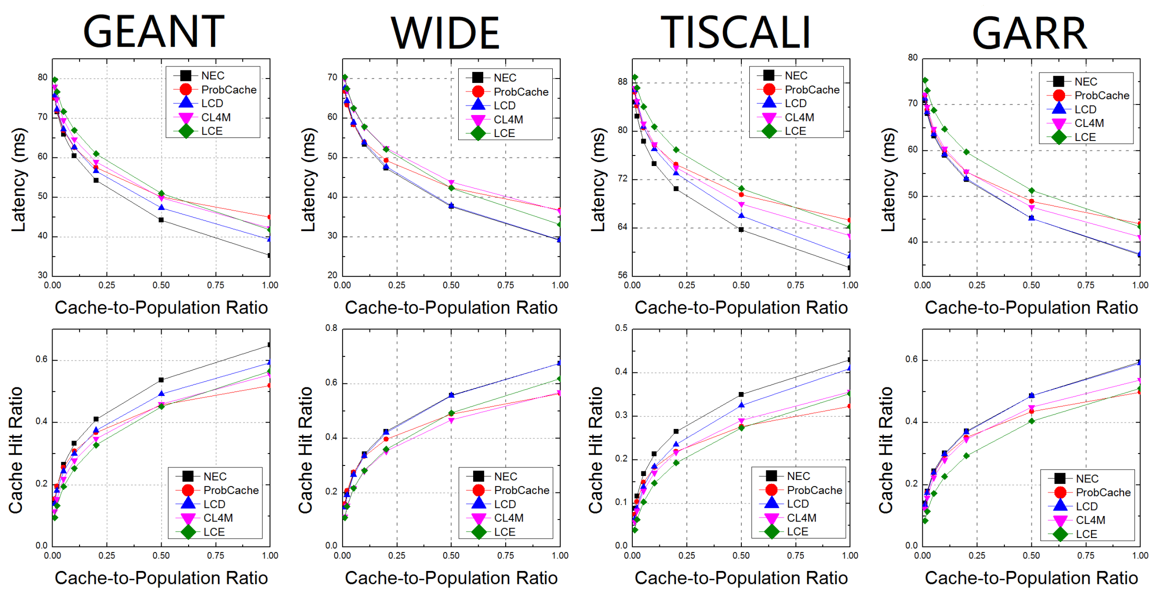


FIGURE 11. Comparison of cache-hit ratios and latencies with cache-to-population ratios{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1}.

various levels. Therefore, the ratio of the expected value of the number of requests in ProbCache to the expected value of the number of requests in LCE is roughly the reciprocal of the probability in ProbCache. We can find that NEC solves the problem of slow diffusion speed in LCD. In workloads that change over time, LCD will have worse adaptability than NEC and ProbCache. We record the average download latency under different numbers of requests, and the results are shown in Fig.10. It can be found that the performance of the LCD is the worst when the number of requests is small, and NEC and ProbCache perform better because of the faster diffusion rate. NEC outperforms ProbCache by virtue of the content filtering feature. The performance of LCD will gradually exceed ProbCache with the increase of number of requests.

Filtering effect. We hope NEC can keep the content in caches with different edge-degrees according to its popularity which means each router needs to perform like a Low-pass filter and most popoular content can be reserved without unpopular ones. We perform experiments in a binary tree with depth of four and measure the number of hits for content of different popularity levels in nodes with different edge-degrees(0/1/2/3). As shown in the Fig.7, top popular content can get the higher hit rate at the edge nodes in NEC than ProbCache, and so is the total hit rate. In the nodes near the center of network(nodes with edge-degree 3), popular content is also hit with a high probability in ProbCache. This is because the edge nodes have frequent cache replacements in ProbCache so the popular content with highest hit rate can wander between different levels of cache. There are more hits and fewer wrong replacements in NEC, so the curve of NEC is smoother. Therefore, NEC has formed a more perfect low-

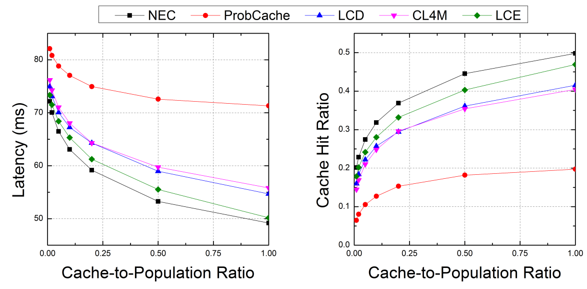


FIGURE 12. Comparison of on-path strategies under the temporal popularity workload.

pass filtering structure than ProbCache, so as to make the most popular content closer to the users.

C. COMPARISON WITH STATE OF ART

We will compare the caching performance of NEC to LCD, ProbCache, CacheLessForMore(CL4M) and LCE in different scenarios. We select the average download latency and cache hit ratio as the evaluation metrics. Latency refers to the time interval between when content request is sent and data packet arrives while cache hit ratio refers to the proportion of content requests served by cache before arriving at source server. The former reflects the the traffic cost and user experience and the latter indicates the server load. Higher cache hit ratio brings lower server load, and lower latency means lower traffic cost.

Comparison in Real-world ISP Topology We first compare the performance of each strategy under different topologies with stationary popularity workload. The results of simulation are shown in Fig.11. It can be found that in GEANT and TISCALI, NEC can significantly reduce the user's download latency and has the highest cache hit rate among all the strategies. With the increase of cache size, advatages of NEC grow and our strategy can reduce the average access latency by 10.2% and increase the cache hit ratio by at most 9.6% when total cache-to-population ratio is 100%. NEC also outperforms the degree centrality based caching strategy CL4M for the reason that CL4M tends to keep contents in the nodes with higher betweenness centrality, which leads to low diffusion rate. CL4M not only has the same problem with LCD but also causes unbalanced caching load in the network because some nodes with small betweenness centrality along the route will hardly have chance to cache content. Although in WIDE and GARR, the performance of NEC is similar to LCD, it is still better than the other strategies. This is because the size of both WIDE and GARR network is small, and the length of the content request path in the network is short(no more than five hops). The advantages of NEC's rapid diffusion cannot be exerted. In a small network, the function of NEC will degenerate to LCD, and the performance difference between strategies will be small.

Comparison in Real-world Traffics Fig.12 presents the comparison for all the strategies under the temporal popularity workload of YouTube traces from the campus network. It can be found that the performance of ProbCache is poor, which is obviously weaker than other performances. This is because the content popularity in Youtube's traffic has time locality, which means popular content may be accessed multiple times in a short period, and the number of requests will be significantly reduced in the future. Compared with other caching strategies, ProbCache's probabilistic caching method cannot always guarantee to keep popular content. In the previous stationary workload, LCE always has a poor performance because it generates excessive redundancy, but this also results in LCE having better adaptability to time-varying traffics. It can be seen that the performance of LCE is significantly better than LCD and CL4M, because these two strategies are difficult to cache content to the edge of the network in time-varying traffic. NEC has better performance because it can cache popular content to the edge of the network in the short time, while avoiding redundant replicas occupying the cache space.

D. STRATEGY IMPLEMENT

To actually deploy a caching strategy into an ICN router, the line speed requirement must be taken into account. Many strategies based on content popularity predict the popularity of content and then make caching decisions in a priori way which causes huge memory overhead and processing delay. We choose two of the simplest popularity-based caching stratgies DBF and MPC for comparison(the implementation of our strategy is based on the cache architecture proposed in [35], and the corresponding results are also given in the article). As the content space grows, the memory overhead for a single content router is shown in the Fig.13. Popularity based strategies need to allocate a counter for every passed-by content item to record the request times. But NEC does not explicitly record the request times of content. So the memory consumption for content information in NEC is only related with the caching size of content router while consumption in popularity based strategies grows with content scale rises. In addition, we implemented a strategy prototype in Xeon(R) E5-2620 v2 CPU (2.10 GHz × 2 CPU cores) which functions as an ICN software router, and we keep sending data packets to test the processing delay and the throughput performance. The number of CPU clock cycles required for processing a single data packet is shown in the Fig.14. FWD is a reference which is the CPU clock cycle consuming by forwarding in a high-speed software NDN router [36]. The throughput performance is shown in the Fig.15. Compared with the popularity based strategies, our strategy has no performance bottleneck under the capability of current devices. It can be determined that our strategy will have better scalability and lower cost when deployed in an ICN content router.

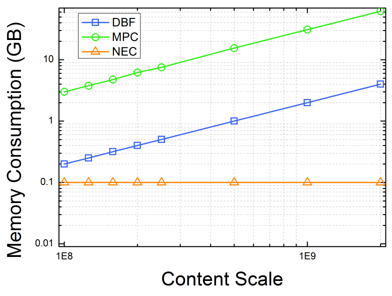


FIGURE 13. Memory consumption with increase of content scale.

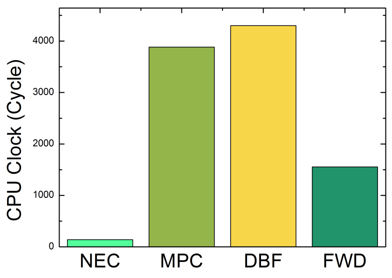


FIGURE 14. CPU-cycle consumption comparison.

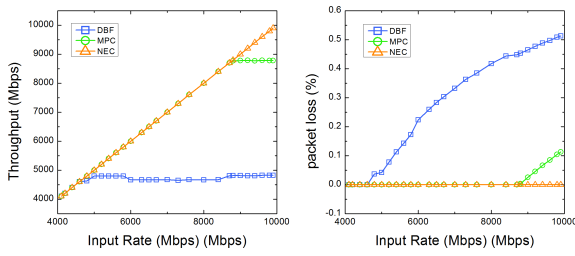


FIGURE 15. Throughput and packet loss rate comparison of different mechanisms in on-path routers.

VI. CONCLUSION

This paper proposes a lightweight caching strategy based on the network location information of routers. This strategy does not need to know the popularity of the content in advance. In essence, NEC provides a way to identify the content popularity by the edge-degree of the node where the content is located. Unpopular contents are filtered in the nodes far away from the edge, and the popular content is gradually distributed to the edge. Compared with existing location-based filtering strategies such as LCD and CL4M, NEC strategy can distribute popular content to the edge faster, with lower caching redundancy and better overall performance. Compared with the strategies based on the popularity of content, NEC is more lightweight and can be easily deployed in the current architecture of ICN routers. The contribution of NEC strategy is to introduce the edge-degree as a metric for decision-making. The number of metrics for the network node location is far from one. We hope that this paper can provide ideas for more researches on lightweight caching strategies in the future.

REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard , "Networking Named Content," in Proc. the 5th international conference on Emerging networking experiments and technologies. Rome, Italy, 2009, pp.1–12.

[2] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Habbal, "Caching in Information-Centric Networking: Strategies, Challenges, and Future Research Directions," IEEE Communications Surveys Tutorials, vol. 20, no. 2, pp. 1443–1474, 2018, doi: 10.1109/COMST.2017.2787609.

[3] G. Zhang, Y. Li, T. Lin et al., "Survey of in-network caching techniques in information-centric networks," Journal of Software, vol.25, no.1, pp.154-175, 2014.

[4] Ding Li, Wang Jinglin, Yang Qifeng, "A Survey of architectural Design of Single Caching Node for Information-Centric Networking", Journal of Network New Media, vol.8, no.3, pp.1-5, 2019

[5] D. Perino and M. Varvello, "A reality check for content centric networking," in Proc. ACM SIGCOMM Workshop on Information-Centric Networking, Toronto, Ontario, Canada, August 2011, pp.44-49.

[6] Y. Meng, M. A. Naeem, R. Ali and B. Kim, "EHCP: An Efficient Hybrid Content Placement Strategy in Named Data Network Caching," in IEEE Access, vol. 7, pp. 155601-155611, 2019.

[7] T. Do, S. Jeon and W. Shin, "How to Cache in Mobile Hybrid IoT Networks?," in IEEE Access, vol. 7, pp. 27814-27828, 2019.

[8] F. Khandaker, S. Oteafy, H. S. Hassanein, and H. Farahat, "A functional taxonomy of caching schemes: Towards guided designs in information-centric networks," Computer Networks, vol. 165, p. 106937, Dec. 2019, doi: 10.1016/j.comnet.2019.106937.

[9] S. Lorenzo, P. Ioannis and P. George, "Hash-routing schemes for information centric networking," in Proc. the 3rd ACM SIGCOMM Workshop on Information-Centric Networking, Hong Kong, China, 2013, pp.27-32.

[10] T. Mick, R. Tourani and S. Misra, "Muncc: Multi-hop neighborhood collaborative caching in information centric networks," in Proc. the 3rd ACM Conference on Information-Centric Networking, Kyoto, Japan, 2016, pp.93-101.

[11] I. Psaras, WK. Chai, G. Pavlou, "Probabilistic in-network caching for information-centric networks," in Proc. the 2nd ICN Workshop on Information-Centric Networking. Helsinki, 2012, pp.55-60.

[12] L. Nikolaos, C. Hao and S. Ioannis, "The LCD interconnection of LRU caches and its analysis," Performance Evaluation, vol.63, no.7, pp.609-634, 2006.

[13] C. Bernardini, T. Silverston, O. Festor, "Mpc: Popularity-based caching strategy for content centric networks," in Proc. IEEE International Conference on Communications(ICC)., Budapest, Hungary, 2013, pp. 3619–3623.

[14] K. Suksomboon, S. Tarnoi, Y. Ji, M. Koibuchi, K. Fukuda, S. Abe, N. Motonori, et al, "Popcache: Cache more or less based on content popularity for information-centric networking," in Proc. 38th Annual IEEE Conference on Local Computer Networks. Sydney, NSW, Australia, 2013, pp.236–243.

[15] A. Ioannou and S. Weber, "A Survey of Caching Policies and Forwarding Mechanisms in Information-Centric Networking," IEEE Communications Surveys Tutorials, vol. 18, no. 4, pp. 2847–2886, 2016, doi: 10.1109/COMST.2016.2565541.

[16] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in Proc. the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, NY, USA, 1999, pp.126-134.

[17] M. Zhang, H. Luo, H. Zhang, "A Survey of Caching Mechanisms in Information-Centric Networking," IEEE COMMUNICATION SURVEYS & TUTORIALS, vol. 17, no. 3, page. 1473-1499, 2015.

[18] Maggs B M, Sitaraman R K. "Algorithmic nuggets in content delivery," Acm Sigcomm Computer Communication Review, vol. 45, no. 3, pp. 52–66, 2015.

[19] Zhang Guo, Jianhui Zhang, Binqiang Wang, ZHANG Zhen, "On-line Popularity monitoring method Based on Bloom Filters and Hash tables for Differentiated Traffc," China Communications, vol. 6, no. s1, pp.72-86, 2016

[20] M. A. Naeem, M. A. U. Rehman, R. Ullah and B. Kim, "A Comparative Performance Analysis of Popularity-Based Caching Strategies in Named Data Networking," in IEEE Access, vol. 8, pp. 50057-50077, 2020.

[21] Abani N, Braun T, Gerla M, "Proactive Caching with Mobility Prediction under Uncertainty in Information-Centric Networks," Proceedings of the 4th ACM Conference on Information-Centric Networking. ICN '17. Berlin, Germany: Association for Computing Machinery, 2017: 88–97.

[22] K. Qi, S. Han and C. Yang, "Learning a Hybrid Proactive and Reactive Caching Policy in Wireless Edge Under Dynamic Popularity," in IEEE Access, vol. 7, pp. 120788-120801, 2019.

[23] Y. Wang, Y. Yang, C. Han, L. Ye, Y. Ke and Q. Wang, "LR-LRU: A PACS-Oriented Intelligent Cache Replacement Policy," in IEEE Access, vol. 7, pp. 58073-58084, 2019.

[24] Anshuman Kalla, Sudhir Sharma, "Exploring off-path caching with edge caching in Information Centric Networking," in Proc. 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), New Delhi, India, 2016.

[25] T. Zhang, X. Fang, Y. Liu and A. Nallanathan, "Content-Centric Mobile Edge Caching," in IEEE Access, vol. 8, pp. 11722-11731, 2020.

[26] A. Mahanti, Carey Williamson, and Derek Eager, "Traffic analysis of a web proxy caching hierarchy," IEEE Network Magazine, vol. 14, no. 3, 2000, pp.16–23.

[27] Wei Koong Chai, Diliang He, Ioannis Psaras and George Pavlou, "Cache "Less for More" in Information-centric Networks," in Proc. NETWORKING 2012, Prague, Czech Republic, May 2012, pp. 27–40.

[28] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in Proc. IEEE INFOCOM Workshops, Orlando, FL, USA, 2012, pp.280-285.

[29] Ding L, Wang J, Sheng Y, et al. "A Split Architecture Approach to TerabyteScale Caching in a Protocol-Oblivious Forwarding Switch," IEEE Transactions on Network and Service Management, 2017, 14(4): 1171–1184.

[30] Song H. "Protocol-oblivious forwarding: unleash the power of sdn through a future-proof forwarding plane," in Proc. the 2nd ACM SIGCOMM workshop on Hot topics in software defined networking. 2013, pp.127–132.

[31] "Information centric networking research group (icnrg)", https://datatracker.ietf.org/doc/draft-irtf-icnrg-ccnxmessages/

[32] L. Saino, I. Psaras and G. Pavlou, "Icarus: A caching simulator for information centric networking(ICN)," in Proc. ICST SIMUTOOLS, Lisbon, Portugal, 2014, pp.66-75.

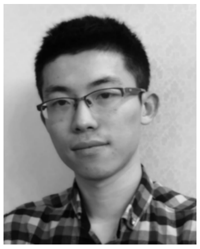
[33] Bahat O , Makowski A M, "Optimal Replacement Policies for Non-Uniform Cache Objects with Optional Eviction," in Proc. IEEE INFOCOM'03, San Franciso, USA, 2003, pp.427-437.

[34] Zink, Michael, Kyoungwon Suh, Yu Gu, and Jim Kurose. "Watch global, cache local: YouTube network traffic at a campus network: measurements and implications." In Multimedia Computing and Networking 2008, vol. 6818, p. 681805. International Society for Optics and Photonics, 2008.

[35] Qifeng Yang, Haojiang Deng and Lingfang, "An Almost-zero Latency Lightweight Mechanism for Caching Decision in ICN Content Router," in Proc. IEEE 38th International Performance Computing and Communications Conference (IPCCC), London, United Kingdom, October 2019.

[36] K. Taniguchi, J. Takemasa, Y. Koizumi, T. Hasegawa, "A Method for Designing High-speed Software NDN Routers," in Proc. the 3rd ACM

Conference on Information-Centric Networking. Kyoto, Japan, 2016, pp.203-204.



QIFENG YANG was born in Yancheng, Jiangsu, China, in 1993. He received the B.S. degree from Peking University, in 2015. He is currently pursuing the Ph.D. degree with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, China. His current research interest includes future networking and in-network caching.

HAOJIANG DENG received the B.S. degree from Wuhan University in 1993, the M.S. degree from the Lanzhou Institute of Physics in 1998, and the Ph.d degree from Institute of Semiconductors of CAS in 2001. He was engaged in the post-doctorate research in Institute of Acoustics (IOA) of CAS till 2003, and began to work in IOA of CAS since 2003. Then he was appointed as a professor in 2007. The main research field of professor Deng covers broadband multimedia communication, digital signal processing in audio and video. He undertook and completed a number of national, provincial and ministerial level scientific research projects.

LINGFANG WANG received the B.S. degree in Computer Science Department of Lanzhou University in 1992, the M.S. degree in Computer Application in Graduate School of Beijing University of Posts and Telecommunications in 2001, and the Ph.d degree from Institute of Acoustics, Chinese Academy of Sciences in 2006. He did post-doctoral research in pattern recognition in the Institute of Automation, Chinese Academy of Sciences.

He is an associate researcher in Institute of Acoustics, Chinese Academy of Sciences. He is an IEEE member, ACM member, a senior member of the Chinese Computer Society, a member of the Acoustic Society, and a professional member of the Chinese Association of System Analysts. He is selected into the National Science and Technology Expert Database and the National Intellectual Property Expert Database.