# CPCache: Cooperative Popularity based Caching for Named Data Networks

Neminath Hubballi and Pankaj Chaudhary
Indian Institute of Technology Indore, India
{neminath, phd2001201004}@iiti.ac.in

***Abstract*—Information Centric Networks improve content delivery performance by caching popular content in the network routers. However, estimating popularity comes with cost of every router exchanging content requests with every other node which leads to a significant overhead. In this paper, we describe `CPCache` which brings cooperation among a group of routers to estimate popularity and make caching decisions. It identifies a set of designated routers in the network to coordinate the popularity estimation and assist edge routers in making caching decisions. In `CPCache`, every edge router associates itself with a designated router from which it seeks assistance to (knowing whether a new content is popular or not) make caching decisions. We describe methods for selecting designated routers, edge router association with a designated router, content popularity estimation mechanism and also performing caching decisions by routers. We evaluate and benchmark the performance of `CPCache` with five other popular caching methods to conclude it offers better performance compared to these methods using simulations done with a popular discrete event simulator.**

***Keywords*—Named Data Network, Content Popularity, Content Caching, Cooperation, Designated Router.***

## I. Introduction

Existing Internet architecture is based on TCP/IP stack and it relies on fetching contents from servers identified by IP addresses. IP addresses locate the network devices. However, with the emergence of multimedia content and evolving user preferences, this architecture faces challenges in meeting demand as it is not well aligned with a data-centric delivery and suffers from inefficient delivery. To tackle the challenges associated with content delivery and improve the user experience with efficient delivery of content, a new networking paradigm known as Content-Centric Networking (CCN) [8] is being evaluated. A notable prototype of CCN is Named Data Networking (NDN) [19]. NDN enables content retrieval based on the content's name, promoting a data-driven approach instead of relying on IP addresses. In NDN, information exchange between consumers and providers is facilitated through Interest and Data packets [19]. Consumers generate Interest packets to request specific content, and in response, the content provider sends a Data packet containing the requested content. Each router processes these packets using three data structures: Content Store (CS) for storing content, Pending Interest Table (PIT) for tracking Interest packets, and Forwarding Information Base (FIB) for routing requests to the next hop router.

NDN facilitates network-level content caching, enabling routers to store consumer-requested content to meet future demands. However, the router caches are limited by capacity constraints. Thus a very limited content can be accommodated in these caches and to maximize the utilization of stored content, they store popular content. Popularity estimation methods enable routers to prioritize frequently requested content in their caches. Routers can estimate popularity individually based on received requests or collaborate with other routers for a collective estimation. However, limiting content caching to only popular items in each router may not be adequate [11], as some requested content, while not highly popular, still contribute to reducing content access time and reducing the load on the server. In order to address the above issues, we introduce the `CPCache`, a demand-driven cooperative popularity estimation approach. It focuses on caching popular content at edge routers, which are within 1-hop distance of consumers, aiming to reduce the content access latency. `CPCache` begins by identifying the most appropriate designated nodes closer to the edge routers based on the network's size. These designated nodes are responsible for tracking both local and global consumer requests via coordination. `CPCache` optimizes computational resources and reduces communication overhead by selectively estimating popularity at designated nodes based on demand. In specific, we make following key contributions this paper.

1. We propose `CPCache` strategy — a demand-driven cooperative popularity estimation technique for caching popular content at edge routers.
2. The `CPCache` strategy uses greedy techniques to determine a set of designated nodes in the network. Further, it allows each edge router to be associated with a designated node to realize cooperative popularity estimation.
3. We assess the performance of the `CPCache` technique using the Icarus [16] simulator with standard network topologies. Our evaluation demonstrates improvements in cache hit ratio and a reduction in content access time compared to state-of-the-art approaches.

We organize the rest of this paper as follows. In Section II we provide a brief overview of related in-network caching work. In Section III we present the working of proposed `CPCache` method. Following that, Section IV presents the simulation results. Finally, Section V concludes the paper.

## II. Literature Review

In order to better utilize the limited cache capacity of NDN routers, different caching techniques [2], [5], [6], [7], [12] have been proposed. These techniques are mainly of two types: On-

path and Off-path techniques, as detailed below.

`(i) On-Path Caching`: On-path caching is a straightforward technique that involves caching and retrieving content from routers located along the content delivery (possibly shortest) path. This method reduces coordination overhead but it is ignorant of the content availability in neighboring (other than on-path) routers. Following are some of the major caching methods of this type.

*Leave Copy Edge (LCEdge) [16]:* Content is only cached at an edge router which is directly connected to the consumer. This method reduces content access time for cached content. However, it reduces the cache hit ratio as it does not use the available space in other routers.

*Leave Copy Everywhere (LCE) [10]:* Each router along the content delivery path caches all content passing through it individually. LCE reduces communication overhead but increases content redundancy due to caching the same content at multiple routers in the network, reducing diversity.

*Prob(p) [9]:* By caching content with a fixed probability value $p$, this technique reduces content redundancy caused by the LCE approach.

*Leave Copy Down (LCD) [9]:* Content is cached at a router one hop down from the provider. LCD improves caching performance by bringing content closer to the consumer with each subsequent request.

*ProbCache [12]:* This strategy caches content in routers located along the delivery path, considering the path's cache capability and cache weight.

*DPCP [18]:* DPCP considers the dynamic popularity of content and facilitates coordination between on-path routers for content caching.

Zheng et al. [20] proposed a dynamic popularity-based caching technique to cache popular content at the most important node along the content delivery path.

`(ii) Off-Path Caching`: In off-path techniques, content is cached and retrieved from any router in the network. This can be achieved either by electing a centralized node [14] or through predefined rules [4], [15] to explore the off-path routers. While off-path techniques enhance the cache hit ratio by utilizing content available in the neighborhood, they also lead to an increase in communication overhead in the network due to the coordination required.

Saino et al. [15] introduced a Hash-based routing technique for retrieving and caching content at designated routers. In this approach, edge routers calculate the hash value of the requested content to locate the designated cache router. While this technique reduces content redundancy and improves the cache ratio, it may lead to increased content access time depending on the distance between the consumer and the designated cache router. The authors of [17] introduced a dynamic clustering strategy for retrieving content from neighboring off-path or on-path routers. Recent works in [4], [5], [13] have designed cooperative caching techniques to leverage content from the off-path neighborhood.

Both caching methods can take popularity into account for their caching choices. We derive motivation from these works and present `CPCache`, which is a cooperative demand driven popularity based caching method. `CPCache` limits popularity estimation only to a subset of nodes and reduces overhead.

## III. CPCache Design and Working Methodology

In this section, we present the design overview of the `CPCache` and provide a detailed explanation of its functioning.

**I) Motivation:** Caching frequently requested content closer to consumers increases performance with reduced access latency and reduces network traffic, as well as the load on the server. Content popularity can be estimated either by having each router maintain a local table to track content requests or, alternatively, only edge routers can hold this information. In both cases, routers do not consider the global request pattern for caching decisions. On the other hand, estimating global popularity requires sharing information related to popularity periodically among routers, resulting in communication overhead. To address this, we design a caching technique that collaboratively assesses local and global content popularity through a set of designated nodes. Our approach begins by identifying *designated nodes* in the network that are responsible for maintaining, estimating and coordinating the content popularity. Each edge node identifies the nearest designated node to update requests and also to make caching decisions. This approach greatly reduces the data exchange by limiting popularity exchange only to a set of designated nodes.

The working of `CPCache` involves many functionalities, as elaborated below.

**II) Selection of Designated Nodes:** `CPCache` limits the popularity estimation and coordination only to designated routers. An important consideration is the selection of these nodes. We use a greedy technique for this purpose which selects nodes with high degree centrality as candidates for designated nodes. Algorithm 1 outlines the procedure for identifying the designated nodes in the network topology G. It begins by sorting the non-edge routers of G in descending order of their degree centrality (higher degree centrality router first). From this list S, it chooses the top N% of routers (routers with a higher degree of centrality in S), relative to the total number of nodes in the topology, as potential candidates for designated nodes. The first router from set N, possessing the highest degree centrality value, is marked as a designated node and added to set `DNodes` (line 7). Subsequently, it selects that node which is at least K hops away (lines 8 to 16) from the existing set of designated nodes. This ensures that these nodes are well spread and not closer to the existing set of designated nodes. We select the parameter K by dividing the diameter d of the topology (representing the maximum distance between any pair of nodes in the network) with the size of set N (number of such nodes required). At each step, the newly selected designated node is added to the set `DNodes`.

For instance, consider the reference topology shown in Fig. 1, assuming K to be 2, and we have a set N containing $\{R_2, R_3, R_7\}$, the process unfolds as follows: $R_2$ is initially added to the `DNodes` set due to its higher degree centrality value. Next, $R_3$ is evaluated, but as it is not at a distance of 2 from $R_2$, it isn't selected as a designated node. Subsequently, we have $R_7$, which is at 2 hops distance from the previously designated $R_2$, making it eligible to be added to the `DNodes` set. As a result, the final `DNodes` set consists of $\{R_2, R_7\}$.

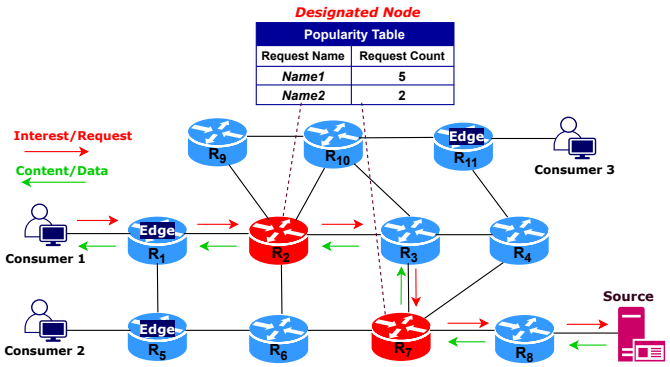



Fig. 1: CPCache Reference Architecture

**Algorithm 1:** Selection of Designated Nodes

**Input:** $G(V, E) \leftarrow$ Network Topology
**Output:** DNodes $\leftarrow$ Set of Designated Nodes
**Initialize:** DNodes $\leftarrow \{\}$
$Z \leftarrow$ FindNonEdgeRouters(G)
$S \leftarrow$ SortRoutersByDegreeCentrality(Z)
$d \leftarrow$ FindDiameterOf(G)
$N \leftarrow$ GetTopNodes(S) // Select top $N$% nodes
$K \leftarrow \frac{d}{N}$ // Distance between designated nodes
DNodes $\leftarrow$ DNodes $\cup N_1$
**for** $N_2$ *to* $N_n$ *in* $N$ **do**
  **for** $X_i$ *in DNodes* **do**
    $Dist \leftarrow$ ShortestPath$(X_i, N_i)$
    **if** $Dist \leq K$ **then**
      Go to 8; Continue with next node in $N$
    **end**
  **end**
  DNodes $\leftarrow$ DNodes $\cup \{N_i\}$
**end**

**III) Associating Edge Router with a Designated Node:** Once the designated nodes are identified, each edge router is required to establish an association with one of the nearest designated nodes to facilitate the sharing of popularity information and making caching decisions. Each edge router $R_i$ selects its designated node $X_i$ from the set DNodes which is closer to it (either number of hops or latency, etc). In this work, we use a shortest path algorithm to choose one as detailed in Algorithm 2.

**IV) Popularity Estimation:** CPCache also uses popularity to guide caching decisions of edge routers and also to coordinated popularity estimation as mentioned earlier. The popularity of a content is determined by the frequency of consumer requests for that particular content within the network. The more frequently an item is requested, the more popular it is considered. We maintain a Popularity Table (PT) to keep track of the request frequencies, which are managed by the designated nodes in the network. These designated nodes collaborate with other designated nodes within the network to estimate the global popularity of content by sharing their local information. When an edge router receives a request for

**Algorithm 2:** Discovering Designated Node for Edge Router $R_i$

$R_i \leftarrow$ Edge Router
DNodes $\leftarrow$ Set of Designated Nodes
$X_i \leftarrow$ Designated Node for $R_i$
$MinDist \leftarrow \infty$
**for** each $X_i \in$ DNodes **do**
  $Dist \leftarrow$ ShortestPath$(R_i, X_i)$
  **if** $Dist < MinDist$ **then**
    $MinDist \leftarrow Dist$
    Pair $X_i$ and $R_i$ // Designate $X_i$ for $R_i$
  **end if**
**end for**

a particular content, it forwards this request to the designated node responsible for updating the popularity table. If an entry for the content already exists in the PT, the request count is incremented. Otherwise, the designated node creates a new entry for the request. Our CPCache strategy employs a demand-driven popularity estimation method, which calculates popularity for the requested content, in contrast to the methods that periodically estimate popularity. This approach is more resource-efficient as it focuses computational efforts on actively considered caching content. Algorithm 3 details how each designated node keeps track of request frequencies. The designated node uses Equation 1 to assess the popularity level of each consumer-requested content. It uses the predefined threshold $\theta$ to determine whether a specific requested item from the edge router is considered popular or not. If the estimated popularity reaches the threshold $\theta$, the content is deemed popular and eligible for caching. The global popularity is estimated as weighted sum of local and global popularity as in Equation 1.

**Algorithm 3:** Updating the Local Popularity of Content

$PT \leftarrow \{\}$ // Initialize Popularity Table
I $\leftarrow$ Index to update request count
$R_i \leftarrow$ Edge Router
$X_i \leftarrow$ Designated Node for $R_i$
Name $\leftarrow$ Content Request at $R_i$
Send Request to $X_i$ // To Update Popularity Table
**if** Name $\in PT$Of$(X_i)$ **then**
  $I_{Name}++$ // Update Count for Name
**else**
  $I_{Name} = 1$ // Create Entry for Name in $PT$
**end if**

$$P_{Name} = (1 - \omega) \times P_L(Name) + \omega \times P_G(Name) \quad (1)$$

In Equation 1, $P_{Name}$ represents the overall popularity of content Name, which is influenced by its demand in both local (all edge routers associated with that designated router) and global regions. $P_L(Name)$ is the local popularity, $P_G(Name)$ is the global popularity of content Name, and $\omega \in [0, 1]$ is the weight parameter set to 0.125 to prioritize local popularity over global popularity, which can help to enhance content retrieval times by ensuring that highly popular content in specific

regions is readily accessible to consumers in those areas. For example, let's consider the reference topology depicted in Fig. 1. In this topology, $R_2$ and $R_7$ are designated nodes and are responsible for maintaining the popularity table. Hence, these nodes record the popularity of content received at edge routers. When $R_1$ receives a request for content `Name`, it forwards the requested content to its designated node, $R_2$, to update the popularity information.

**V) Content Caching by a Router:** Caching decisions are made when a content provider responds with a Data packet corresponding to the request. In this process, each intermediate router between the content provider and the consumer decides which content to store in the CS (cache) and which to evict. The `CPCache` strategy facilitates cooperative caching at the consumer edge router through consultation with its designated node. When router $R_i$ receives a Data packet and has sufficient space in CS, it always caches the content. This approach aims to maximize cache utilization and promote a faster converging hitting rate. When the cache capacity of the router $R_i$ is full, the caching decisions made by it are differ depending on whether $R_i$ is an edge router or an intermediate router. If $R_i$ is not an edge router, it will simply evict content from its CS using the Least Recently Used (LRU) policy and cache the newly arrived content. On the other hand, if $R_i$ is an edge router, then the caching decision is influenced by the popularity of the newly arrived content. Edge router $R_i$ consults its respective designated node to obtain the popularity information of the new content. If the popularity of the new content surpasses a predefined threshold, it caches it in its CS by replacing the least recently accessed content using LRU. Otherwise, it ignores the new arrival. Placing highly popular content at consumer edge routers helps reduce content retrieval times. Algorithm 4 outlines the details of the content caching process. In the reference topology shown in Fig 1, if content `Name1` arrives at full-capacity edge router $R_1$, $R_1$ collaborates with its designated node $R_2$ to decide whether to cache the content based on its popularity; if it is considered popular, cache it; otherwise, ignore it.

**Algorithm 4:** Caching Content at a Router $R_i$

```
Name ← New Content at R_i
Size ← SizeOf(Name)
C_i ← CapacityOf(R_i)
O_i ← OccupancyOf(R_i)
if Name ∉ CacheOf(R_i) then
    if C_i − O_i ≥ Size then
        Cache Name at R_i //Maximize Cache Utilization
    else if R_i is an Edge Router then
        Cache Name at R_i if P_Name ≥ θ // Keeps Popular Content
    else
        Cache Name at R_i //Use LRU Policy
    end if
end if
```

## IV. Experiments and Evaluation

This section begins with an overview of the simulation setup. Following that, we introduce the evaluation metrics and network topologies used for comparing the performance of the proposed `CPCache` against LCEdge [16], LCE [10], Prob(p=0.5) [9], ProbCache [12], and DPCP [18]. Subsequently, we present the simulation results.

**I. Simulation Setup:** We utilize the Icarus [16] simulator, a commonly used tool for evaluating caching techniques within the ICN/NDN framework, to assess the performance of our proposed `CPCache` caching method. Icarus is developed in Python and easily adaptable to various ICN architectures, offering a wide range of experimental configurations and standard network topologies, providing an extensive platform for evaluating caching methods. In Icarus, the simulation workload is modeled by Poisson and Zipf [3] distributions, which capture the arrival of consumer requests and adjust the different levels of content popularity. In our simulations, we utilize these distribution functions to generate $10^5$ requests from a content universe containing $10^4$ distinct items, all of which are located at a single content source. We set the content arrival rate $\lambda$ to 10 requests per second, which means, on average, ten requests come in every second, and the Zipf popularity parameter $\alpha$ to 0.7 and 0.8 to replicate varying degrees of content popularity. We assume that all routers in the network topologies are capable of caching content, with each router having a uniform capacity ranging from 0.1% to 0.5%, and all content items are of equal size. This capacity, in comparison to the size of the content universe, is relatively small. In our simulations, all caching techniques employ the best-path strategy to route packets toward the content provider, and each strategy utilizes the LRU policy to remove content when a router's capacity is reached. The link latency between each pair of connected nodes is set at 5 milliseconds. For the `CPCache` strategy, the default popularity threshold $\theta$ value is fixed at 50, and the value of N is set to 5% of the total nodes in the topology for identifying the designated nodes. The simulations are run five times, and the average results from these runs are used to generate the graphs showing the performance. The details of the simulation parameters used for the experiments are presented in Table I.

TABLE I: Simulation Settings

| Parameters | Value |
|---|---|
| Network topology | GEANT and TREE |
| Content universe (catalog) size | $10^4$ objects |
| Number of requests | $10^5$ objects |
| Number of warm-up requests | $10^3$ objects |
| Content request arrival rate | Poisson distribution, $\lambda$ = 10 requests per second |
| Content popularity model | Zipf, $\alpha \in [0.7, 0.8]$ |
| Routers cache capacity | [0.1 to 0.5]% of content catalog |
| Replacement Strategy | LRU |
| Experiment repetitions | 5 |

**II. Evaluation Metrics:** We used two standard evaluation metrics, namely cache hit ratio and content access time, to assess the effectiveness of our proposed technique.

*1. Cache Hit Ratio:* This metric reflects the proportion of content served directly from the routers' caches. A higher ratio indicates a lighter load on the content server.

*2. Content Access Time:* This metric measures the collective

time taken for all consumer requests to reach the content provider and for the corresponding content to travel back to the consumers.

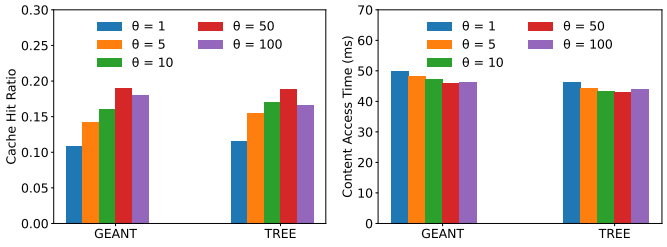
**III. Network Topology:** We conducted simulations using two standard network topologies with distinct sizes and structures: GEANT [1] and TREE. GEANT is a high-speed data communications network connecting research and education institutions across Europe. It comprises 40 routers, with 8 of them designated as one-degree routers, acting as edge routers. We assigned artificial consumers to each of these edge routers, and one artificial source node is attached to the router with the highest degree. On the other hand, TREE is an artificial hierarchical network topology available in the Icarus. In this topology, the root node functions as the content source, the intermediate nodes serve as routers responsible for content caching, and the leaf nodes are marked as consumers. The TREE topology is configured with a level value of 6 and a branching factor of 2, resulting in a total of 63 nodes. Table II has more details about these topologies.

TABLE II: Topology Information

| Topology | Node | Link | Consumer | Source | Router | Diameter |
|---|---|---|---|---|---|---|
| GEANT | 49 | 70 | 8 | 1 | 40 | 10 |
| TREE | 63 | 62 | 32 | 1 | 30 | 10 |

**IV. Simulation Results:** Here, we present the simulation results for all six caching techniques conducted on two different network topologies.

**i) Determining the Appropriate Popularity Threshold:** In this section, we first explore the determination of the suitable popularity threshold value $\theta$ through evaluations. This threshold helps determine what content is considered popular within a specific region. Fig. 2 demonstrates the effects on cache hit ratio and content access time while varying the threshold value, with a fixed cache size of 0.3% and a Zipf $\alpha = 0.8$. The results illustrated in Fig. 2 show that setting the $\theta$ at 50 yields the most favorable outcomes in terms of both cache hit ratio and content access time. Consequently, we set the default value of $\theta$ to 50 for all subsequent experiments.

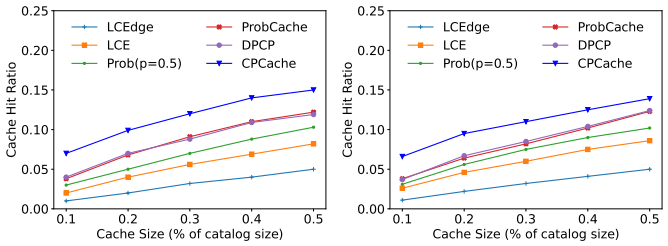



(a) Cache Hit Ratio (b) Content Access Time

Fig. 2: Estimation of Optimal Popularity Threshold ($\theta$) for Various Network Topologies with Fixed Parameters (Cache Size = 0.3%, Zipf $\alpha$ = 0.8).

**ii) Evaluating the Influence of Cache Size and Zipf $\alpha$:** Here, we present the simulation outcomes for all six caching techniques. We evaluate the performance of `CPCache` by examining cache hit ratios and content access times on GEANT and TREE network topologies under different router cache sizes, ranging from 0.1% to 0.5%, and adjusting the Zipf popularity $\alpha$ to 0.7 and 0.8.

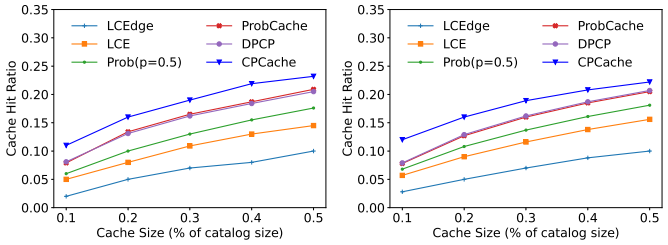
Fig. 3 illustrates the cache hit ratios for GEANT and TREE topologies with Zipf $\alpha$ set at 0.7. As shown in Fig. 3, an increase in router cache capacity results in higher cache hit ratios for all the techniques, indicating that a greater proportion of consumer requests are being satisfied by the router's cache. Notably, regardless of the cache size, our proposed `CPCache` strategies consistently outperform the other five caching techniques on both topologies. On GEANT topology, the `CPCache` achieves up to a 75% higher cache hit ratio than DPCP (second best strategy) for smaller cache sizes (i.e., 0.1%) and up to 26.05% for larger cache sizes (i.e., 0.5%). Similarly, on TREE topology, the `CPCache` achieves up to a 73.6% higher cache hit ratio than DPCP for smaller cache sizes (i.e., 0.1%) and up to 12.1% for larger cache sizes (i.e., 0.5%).




(a) GEANT (b) TREE

Fig. 3: Cache Hit Ratio for Different Cache Sizes on Different Network Topologies ( Zipf $\alpha$= 0.7).

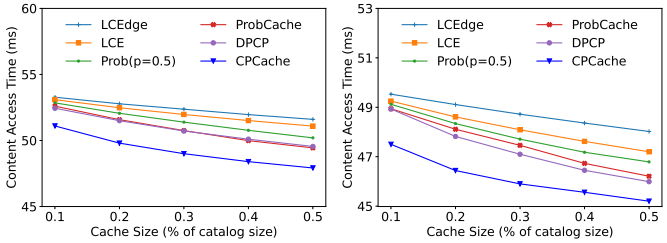
Fig. 4 presents the cache hit ratio for both GEANT and TREE topologies with a Zipf $\alpha$ value of 0.8. Increasing the $\alpha$ parameter results in improved cache hit ratios for all six caching methods. Figs. 4a and 4b demonstrate that the `CPCache` consistently outperforms other caching techniques, regardless of cache size and Zipf alpha parameter. For GEANT topology, the cache hit ratio of the `CPCache` surpasses DPCP by up to 35.8% with a cache size of 0.1% and up to 13.2% with a cache size of 0.5%. On TREE topology, the cache hit ratio of the `CPCache` outperforms DPCP by up to 51.9% with a cache size of 0.1% and up to 7.25% with a cache size of 0.5%.




(a) GEANT (b) TREE

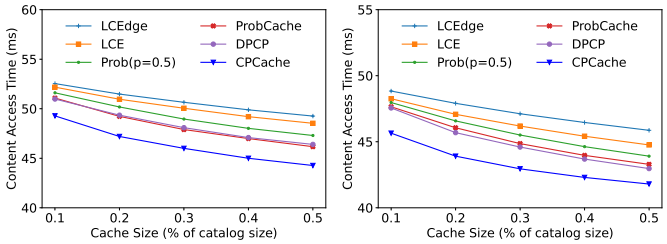
Fig. 4: Cache Hit Ratio for Different Cache Sizes on Different Network Topologies ( Zipf $\alpha$= 0.8).

Fig. 5 presents the content access time for GEANT and TREE topologies with a Zipf $\alpha$ value of 0.7. It is evident that, as router capacity increases, the content access time for all techniques decreases, indicating that most consumer requests are served by the cache of the nearest router. On both topologies, our proposed CPCache outperformed the other five caching techniques regardless of cache size. On the GEANT topology, the CPCache exhibits content access times up to 3.4% lower than DPCP. Similarly, on TREE topology, it is up to 2.96% lower than DPCP.




(a) GEANT (b) TREE

Fig. 5: Content Access Time (in milliseconds) for Different Cache Sizes on Different Network Topologies (Zipf $\alpha$= 0.7).




(a) GEANT (b) TREE

Fig. 6: Content Access Time (in milliseconds) for Different Cache Sizes on Different Network Topologies (Zipf $\alpha$= 0.8).

Fig. 6 illustrates the content access time for GEANT and TREE topologies with a Zipf $\alpha$ value of 0.8. As the Zipf $\alpha$ parameter increases, a notable reduction in content access time is observed across all six caching strategies. Figs. 6a and 6b demonstrate the consistent superiority of the CPCache strategy over other caching methods, regardless of cache size and $\alpha$ parameter. On GEANT topology, CPCache achieves a content access time up to 4.58% lower than DPCP. Similarly, on TREE topology, CPCache outperforms DPCP by up to 4.1% in terms of content access time.

## V. Conclusion

In this paper, we presented CPCache a cooperative popularity estimation technique that caches popular content at edge routers through coordination with designated nodes. The performance of CPCache is assessed using the Icarus, a Python-based discrete event simulator on two distinct network topologies. We compared our proposed CPCache technique with five state-of-the-art in-network caching methods (LCEdge, LCE, Prob(p), ProbCache, and DPCP) using two performance metrics: cache hit ratio and content access time. The simulations were carried out by changing the cache size and the Zipf $\alpha$ parameter. The results of the simulations indicate that CPCache reduces content delivery time and increases the cache hit ratio compared to other caching techniques on both topologies. We intend to extend our CPCache approach to enable centralized caching decisions through designated nodes and also evaluate the performance with other parameters like cache diversity.

## References


[1] GEANT. Online. Available: https://geant3plus.archive.geant.net/Pages/home.html. Accessed: 2023-10-28.

[2] S. Alduayji, A. Belghith, A. Gazdar, and S. Al-Ahmadi. Pf-edgecache: Popularity and freshness aware edge caching scheme for ndn/iot networks. *Pervasive and Mobile Computing*, 91:101782, 2023.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. IEEE INFOCOM'99*, volume 1, pages 126–134, 1999.

[4] P. Chaudhary, N. Hubballi, and S. G. Kulkarni. Ncache: neighborhood cooperative caching in named data networking. In *2022 5th International Conference on Hot Information-Centric Networking (HotICN)*, pages 36–41, 2022.

[5] P. Chaudhary, N. Hubballi, and S. G. Kulkarni. encache: Improving content delivery with cooperative caching in named data networking. *Computer Networks*, 237:110104, 2023.

[6] J. Hou, H. Xia, H. Lu, and A. Nayak. A gnn-based approach to optimize cache hit ratio in ndn networks. In *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pages 1–6, 2021.

[7] N. Hubballi, P. Chaudhary, and S. G. Kulkarni. Pepc: Popularity based early predictive caching in named data networks. In *Proc. IEEE Consumer Communications & Networking Conference*, pages 1–6, 2024.

[8] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proc. ACM CoNEXT*, pages 1–12, 2009.

[9] N. Laoutaris, H. Che, and I. Stavrakakis. The lcd interconnection of lru caches and its analysis. *Performance Evaluation*, 63(7):609–634, 2006.

[10] N. Laoutaris, S. Syntila, and I. Stavrakakis. Meta algorithms for hierarchical web caches. In *Proc. IEEE Int. Conf. Perform. Comput. Commun. (IPCCC)*, pages 445–452, 2004.

[11] M. D. Ong, M. Chen, T. Taleb, X. Wang, and V. C. Leung. Fgpc: Fine-grained popularity-based caching design for content centric networking. In *Proc. 17th ACM Int. Conf. Model. Anal. Simulat. Wireless Mobile Syst.*, pages 295–302, 2014.

[12] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *Proc. 2nd Ed. ICN Workshop Inf.-Centric Netw. (ICN)*, pages 55–60, 2012.

[13] A. Reshadinezhad, M. R. Khayyambashi, and N. Movahedinia. An efficient adaptive cache management scheme for named data networks. *Future Generation Computer Systems*, 148:79–92, 2023.

[14] J. Rihab and L. C. Fourati. Ccnflow: Content-centric networking managed by openflow controller. In *Proc. ComNet, 2018*, pages 1–5, 2018.

[15] L. Saino, I. Psaras, and G. Pavlou. Hash-routing schemes for information centric networking. In *Proc. ACM SIGCOMM ICN Workshop*, pages 27–32, 2013.

[16] L. Saino, I. Psaras, and G. Pavlou. Icarus: a caching simulator for information centric networking (icn). In *Proc. 7th Int. Conf. Simul. Tools Techn.*, pages 66–75, 2014.

[17] M. Yoshida, Y. Ito, Y. Sato, and H. Koga. Performance evaluation of popularity-aware dynamic clustering scheme for distributed caching in icn. In *APSIPA ASC*, pages 185–190, 2022.

[18] M. Yu and R. Li. Dynamic popularity-based caching permission strategy for named data networking. In *Proc. 22nd CSCWD'18*, pages 576–581, 2018.

[19] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[20] Q. Zheng, Y. Kan, J. Chen, S. Wang, and H. Tian. A cache replication strategy based on betweenness and edge popularity in named data networking. In *Proc. IEEE ICC*, pages 1–7, 2019.