An Efficient Content Retrieval and Content Placement Approach for Named Data Networks

Matta Krishna Kumari
CSE Group
IIIT Sri City
Tirupati, India
krishnakumari.m@iiits.in

Nikhil Tripathi
CSE Group
IIIT Sri City
Tirupati, India
nikhil.t@iiits.in

***Abstract*—Named Data Network (NDN) is the future-generation Internet architecture proposed to address the issues in the current Internet architecture (TCP/IP) such as high content access latency, single point of failure, etc. NDN supports in-network caching that significantly enhances the network performance and facilitates scalable content distribution. However, the state-of-the-art in-network caching approaches suffer from drawbacks such as high lookup repetition overhead, poor cache utilization, and high content redundancy. To overcome these issues, in this paper, we propose a new content retrieval and content placement approach for NDN. The proposed approach reduces the lookup repetition overhead by minimizing the number of router consultations required for content retrieval. Moreover, the proposed approach improves cache utilization and reduces content redundancy by optimally placing the content on the most suitable router. The experimental results show that this approach improves the overall performance of the NDN architecture in terms of both content access latency and Cache Hit Ratio (CHR). We also compare the performance of the proposed approach with state-of-the-art approaches in a real-world topology and show that it outperforms the previously known approaches.**

***Index Terms*—NDN, ICN, future Internet architecture, in-network caching**

I. INTRODUCTION

THE number of Internet users has grown significantly in recent years due to emerging domains such as the Internet of Things (IoT), vehicular networks, etc [21]. This advancement exceeded the capabilities of the Internet as the underlying TCP/IP architecture is designed for a limited number of hosts [1]. Moreover, the current architecture is a host-centric approach that leads to problems such as higher bandwidth consumption, content access latency, single point of failure, etc [2]. To address these challenges, researchers introduced a next-generation Internet Architecture called Named Data Network (NDN) [2]. The working of this architecture is based on the fact that the users are interested in accessing the data with minimal latency without bothering about the data source. The prime advantage of NDN over current Internet architecture is its ability for in-networking caching [3]. This caching approach allows intermediate routers to store content in their local caches called $Content\ Store$ (CS). This greatly improves network performance by reducing content access delay and optimizing bandwidth utilization. Furthermore, the replication of content in caches eliminates the vulnerability to single points of failure.

The performance of NDN architecture is primarily based on three strategies: *content retrieval*, *content placement*, and *content replacement*. The content retrieval strategies involve fetching content from either a router's CS or the content producer itself. The content placement strategies focus on placing the content on the CS of the most appropriate routers to maximize the Cache Hit Ratio (CHR). On the other hand, the content replacement strategies involve removing content from the cache once it is full. Traditional content replacement methods like LFU, LRU, and FIFO are known to give promising results within the NDN context. However, refining the content retrieval and content placement strategies for NDN is still a key concern for the researchers.

In the past few years, researchers have come up with new approaches for content retrieval and content placement in NDN. However, the known approaches suffer from a few common drawbacks such as high lookup repetition overhead, poor cache utilization, high content redundancy, high content access latency, and low CHR. To overcome these drawbacks, in this paper, we propose an efficient content retrieval and content placement approach that not only improves the CHR but also reduces the content access latency. During content retrieval, our strategy minimizes the number of router consultations to reduce the lookup repetition overhead. On the other hand, for content placement, it optimally selects an on-path router. We test the performance of our proposed strategy in a real-world topology and show that it achieves higher CHR and lower content access latency as compared to the previously known state-of-the-art approaches.

The rest of the paper is organized as follows. NDN background is discussed in Section II. We present the literature review in Section III. We present our proposed caching strategy in Section IV. We describe the experiments conducted to test the detection performance of the proposed strategy and the obtained results in Section V. Finally, the paper is concluded in Section VI.

II. BACKGROUND

NDN is a content-centric architecture in the sense that it uses data names for packet forwarding instead of IP addresses

979-8-3503-3094-6/24/$31.00 ©2024 IEEE

[2]. Moreover, it uses a hierarchical naming convention to track and route content [4]. The architecture contains three types of entities - 1) consumers, 2) producers, and 3) routers. The consumer initiates the communication by sending an interest packet for accessing content. The producer creates the content and replies back with data packets to the consumer requesting the content. The routers are responsible for forwarding the interest and data packets to up and downstream interfaces, performing content retrieval, managing the content placement, and executing the content replacement strategies. The routers consist of three essential data structures [5] - 1) $ContentStore$ (CS), 2) $Pending\ Interest\ Table$ (PIT), and 3) $Forwarding\ Information\ Base$ (FIB). The CS is a limited storage space available at a router to accomplish the in-network caching. The CS stores the entry for a received data packet in the format `<content name: data packet>` [6]. The PIT stores an entry for each unsatisfied interest packet in the format `<content name: requested interfaces>` until a data packet corresponding to the requested content is received or the timeout period expires. Meanwhile, in case the router receives more interest packets for the same content, the corresponding PIT entry is updated by appending the interfaces on which the interest packets are received [6]. The FIB stores the entries required to forward interest packets to the next upstream router towards the producer. The entries in FIB are in the format `<content name: upstream interface>` [6].

III. LITERATURE REVIEW

The in-network caching strategies in NDN can be divided into two broad categories - *On-path* and *Off-path* [6]. The on-path is the default caching strategy in NDN [7]. According to this strategy, content retrieval and its caching are done only on those routers that fall along the shortest path from the consumer to the producer. Placing the content on an on-path router has several advantages over off-path placement such as easier implementation, no single point of failure issues, less communication and computational overhead, etc. Thus, in this section, we restrict our discussion to the caching strategies that involve placing the content on the on-path routers only.

Leave Copy Everywhere (LCE) [8] is a classical on-path caching strategy. This strategy suggests caching the content on all on-path routers. However, this leads to poor cache utilization due to significant content redundancy. To address this issue, several works in the literature discuss different improvements over LCE. These works can be divided into different categories [20] which are as follows:

1) Distance-based: A few approaches suggest caching the content in a router's CS only if the router is in close proximity to the consumer [10] or the producer [12]. However, these approaches suffer from drawbacks such as poor cache-hit ratio and/or higher content access latency [12], [13]. Authors in [8] proposed a new variant called Move Copy Down (MCD). MCD also suggests caching the content at the router directly connected to the producer. In addition, with each repeated interest packet, the content is also copied at a router one hop down towards the consumer. The drawback of this approach is that as the number of interest packets increases, the content redundancy also increases. In [6], authors introduced the Neighborhood Cooperative Caching (NCache). This approach suggests content lookup in not only the on-path routers but also the one-hop neighbours of the on-path routers. However, a drawback of this approach is the increased overhead during the content lookup process. This is because it checks a router multiple times for the same content if that router is directly connected to multiple on-path routers.

2) Probability-based: Probability-based caching strategies involve storing content in a router's CS based on the probability assigned to the CS. This probability is either predefined [14] or computed dynamically [15], [16], [17], [18], [6], and [19] based on the router's distance from the producer and the consumer. The strategies that involve assigning fixed probability suffer from drawbacks such as improper cache utilization [20]. Due to this limitation, most of the approaches under this category assign dynamic probabilities to the routers. The dynamic probability-based strategies are known to have better cache utilization with less content access latency [21]. However, these approaches fail to reduce the content redundancy and memory usage [9].

3) Centrality-based: Centrality-based on-path caching strategies suggest caching the content at a router that is either midway on the shortest path from producer to consumer (CL4M [22]) or common for most of the shortest paths existing between the producer and the consumer (CMBA [23]). However, authors in [13] showed that finding such a router in a large topology may result in significant overhead. Also, content redundancy increases when routers share identical centrality metrics [9].

4) Popularity-based: The popularity-based caching strategies suggest caching the content based on its popularity [24], [25], [26], [27], [28], [29], [30], [31], [32], [21], [13], [33], [20]. The content popularity is calculated based on the frequency of its retrieval. Content is cached at a router if its popularity exceeds a threshold value [34], [35], [36], [37], [38]. However, determining the popularity of each content by the individual routers requires significant computational resources [9], [32].

IV. PROPOSED APPROACH

Our proposed strategy operates in two stages - i) content retrieval, and ii) content placement as discussed in the next few subsections.

A. Content Retrieval

Algorithm 1 describes the content retrieval stage of our proposed strategy. To retrieve content from a producer, the consumer first computes the shortest path from itself to the producer. The intermediate routers along this shortest path are known as on-path routers [6]. After computing the shortest path, the consumer sends *interestpacket* to the router directly connected to it (edge router), $\mathcal{E}_C$, towards the producer **(Step 1)**. On receiving *interestpacket*, $\mathcal{E}_C$ performs a series of actions.

Algorithm 1 Content Retrieval Algorithm

```
Input: OnPathRouters, interestpacket
 1: Consumer forwards interestpacket to $\mathcal{E_C}$
 2: if interestpacket.content in $CS$ of $\mathcal{E_C}$ then
 3:     return interestpacket.content
 4: else if interestpacket.content in $RIT$ of $\mathcal{E_C}$ then
 5:     RouterID ← $RIT$[interestpacket.content]
 6:     Path1 ← shortestPath($\mathcal{E_C}$, RouterID)
 7:     Path2 ← shortestPath($\mathcal{E_C}$, producer)
 8:     if len(Path1) ≤ len(Path2) then
 9:         $V_1 \leftarrow \mathcal{E_C}$
10:         $V_2$ ←next on-path router towards RouterID
11:         for ($V_1$, $V_2$) in Path1 do
12:             Forward interestpacket to $V_2$
13:             if $V_2$ == $RIT$.RouterID then
14:                 return interestpacket.content
15:             end if
16:             $V_1 \leftarrow V_2$
17:             $V_2$←next on-path router towards RouterID
18:         end for
19:     else
20:         $V_1 \leftarrow \mathcal{E_C}$ ;$V_2$ ←next on-path router towards producer
21:         for ($V_1$, $V_2$) in Path2 do
22:             OnPathRoutersDegree.append($V_1$, deg($V_1$))
23:             if $V_1 == \mathcal{E_P}$ and $V_2$ == producer then
24:                 Store OnPathRoutersDegree at $V_1$
25:                 return interestpacket.content
26:             end if
27:             Forward interestpacket to $V_2$
28:             $V_1 \leftarrow V_2$
29:             $V_2$ ← next on-path router towards producer
30:         end for
31:     end if
32: else
33:     Execute Steps 20 - 29
34: end if
```

It first checks if the content requested by a consumer is available in its CS **(Step 2)**. If the content is found, $\mathcal{E_C}$ serves it to the consumer **(Step 3)** using *datapacket*. However, if the content is not found, $\mathcal{E_C}$ consults its Pending Interest Table (PIT) to check whether it is a duplicate or unique *interestpacket*. If it is a duplicate *interestpacket*, $\mathcal{E_C}$ discards it and updates its PIT as discussed earlier in Section II. If it is a unique interest packet, $\mathcal{E_C}$ consults its Router Index Table (RIT) to find the router responsible for caching the requested content. RIT is an additional data structure maintained by the $\mathcal{E_C}$, and contains two fields - `ContentName` and `RouterID`. It maps the content to the router responsible for caching that content. If RIT contains an entry for the requested content, $\mathcal{E_C}$ extracts the `RouterID` of the router responsible for caching the requested content **(Step 5)**. Subsequently, $\mathcal{E_C}$ computes the shortest path from itself to the `RouterID` **(Step 6)** as well as to the producer **(Step 7)**. If `RouterID` is found to be closer, $\mathcal{E_C}$ forwards the *interestpacket* to it **(Steps 9 - 18)**. However, if the producer is found to be closer **(Steps 20 - 29)** or if the *interestpacket.content* does not exist in the RIT **(Step 33)**, $\mathcal{E_C}$ crafts an interest packet with one additional field `OnPathRoutersDegree` and populates it with its router ID and its degree **(Step 22)**. Subsequently, $\mathcal{E_C}$ forwards *interestpacket* to V_2 **(Step 27)**. On receiving *interestpacket*, an intermediate router appends its ID and its degree in the field `OnPathRoutersDegree` and forwards it to the next on-path router towards the producer. As soon as *interestpacket* arrives at the on-path router $\mathcal{E_P}$ directly connected to the producer, $\mathcal{E_P}$ temporarily stores the values present in `OnPathRoutersDegree` field of *interestpacket* in a data structure in the format $< Content :$ `OnPathRoutersDegree` $>$ **(Step 24)**, and forwards *interestpacket* to the producer. Subsequently, the producer serves the content to the consumer **(Step 25)**. It is to be noted that the data packet traverses the same path that is traversed by the corresponding interest packet.

B. Content Placement

We propose an efficient content placement approach based on the number of connections of an on-path router. Our content placement approach is described in Algorithm 2. In this approach, $\mathcal{E_P}$ receives *datapacket* from the producer. First, it finds the maximum degree router for *datapacket.content* from its temporary data structure (as mentioned in Step 24 of Algorithm 1) and then adds the ID of that particular router, `RouterID`, in the *datapacket* **(Step 2)**. Subsequently, *datapacket* is forwarded towards $\mathcal{E_C}$ along the same path **(Step 4)**. Every router on the path compares its ID with

Algorithm 2 Content Placement Algorithm

```
Input: OnPathRoutersDegree, datapacket, Path2
 1: $V_1$ ← Producer ; $V_2 \leftarrow \mathcal{E_P}$
 2: $\mathcal{E_P}$ adds RouterID to the datapacket
 3: for ($V_1, V_2$) in Path2 do
 4:     Forward datapacket to the $V_2$
 5:     if $V_2$ == RouterID then
 6:         Place content at $V_2$
 7:         Broadcast ACK packet to all the routers
 8:     else
 9:         $V_1 \leftarrow V_2$ ; $V_2$ ← next on-path Router to $V_1$
10:     end if
11: end for
12: for each $\mathcal{E_C}$ in edge routers do
13:     if ACK Received by $\mathcal{E_C}$ then
14:         if Action == Placed then
15:             $RIT$[datapacket.content] = RouterID
16:         else
17:             del $RIT$[datapacket.content]
18:         end if
19:     end if
20: end for
21: $\mathcal{E_C}$ forwards datapacket to the consumer
```

`RouterID` **(Step 5)**. If both match, the content is placed in that particular router **(Step 6)**. After this, the router broadcasts an ACK packet containing the details - *datapacket.content*, `RouterID`, and `Action` to all the routers **(Step 7)**. Here, `Action` is a binary value that can be either `Placed` or `Evicted`. On receiving this ACK packet, an edge router $\mathcal{E}_{\mathcal{C}}$ updates its RIT according to the `Action` **(Steps 13 - 17)**. If `Action` is `Placed`, $\mathcal{E}_{\mathcal{C}}$ adds an entry in its RIT as shown in **(Step 15)**. However, if `Action` is `Evicted`, $\mathcal{E}_{\mathcal{C}}$ removes the entry from RIT **(Step 17)**. Finally, $\mathcal{E}_{\mathcal{C}}$ forwards *datapacket* to the consumer that requested the content **(Step 21)**.

V. EXPERIMENTS & RESULTS

In this section, we first discuss the setup used to assess the performance of our proposed strategy. Subsequently, we present the experimental results followed by the sensitivity analysis of the proposed approach with respect to a few parameters.

A. Experimental Setup

We conducted the experiments using the Icarus simulator [10] configured on a computer running Ubuntu OS and having a Core i5 processor with 16 GB of physical memory. To compare our strategy with the similar approaches known in the literature, we also implemented those approaches in the simulator. Moreover, we used a real-world topology called Internode network [39] to test the performance of different caching approaches. The topology contains 66 nodes, out of which 25 and 22 were assigned as consumers and producers, respectively. The remaining 19 nodes were assigned as intermediate routers. We used simulation parameters as shown in Table I. In particular, we put 300,000 unique content objects in the network. These objects were requested by the consumers for a total of 600,000 times. The content objects are uniformly distributed among the producer nodes. Moreover, the content requests sent by the consumers are governed by a Poisson distribution-based method. The requests for objects followed a round-robin pattern, ensuring that subsequent requests for the initial content object occur only after all 300,000 content objects have reached the receiving node. The network cache size C is expressed as a fraction of the entire content population that is accessible within the network. The content popularity has been implemented using the Zipf Mandelbrot content distribution parameter α in the range of 0.04 to 0.4 [13]. α captures correlations within content requests, reflecting that recently requested content is likely to be requested again in the near future. Increasing α causes requests with a focus on a smaller subset of content. Additionally, the internal link delay, measured as the delay between receivers and routers, is set to 2ms. The external link delay, which signifies the delay from sources to routers, is specified as 34ms. These values are utilized to compute latency in accordance with prior research findings [11], [21]. It is to be noted that we used the standard Least Recently Used (LRU) policy as the cache replacement method for all experiments.

TABLE I: Parameters and Values

Parameter	Value
Number of content objects	300,000 objects
Number of content requests	600,000 objects
Content request order	Round robin
Request rate	1 request per second
Link delay	Internal: 2ms, External: 34ms
Network cache size	$C \in [0.5 - 3.0]$
Zipf Distribution parameter	$\alpha \in [0.04 - 0.4]$
Cache replacement policy	LRU
Content placement	Uniform
Workload	Trace-driven
Number of times experiments conducted	5

B. Results

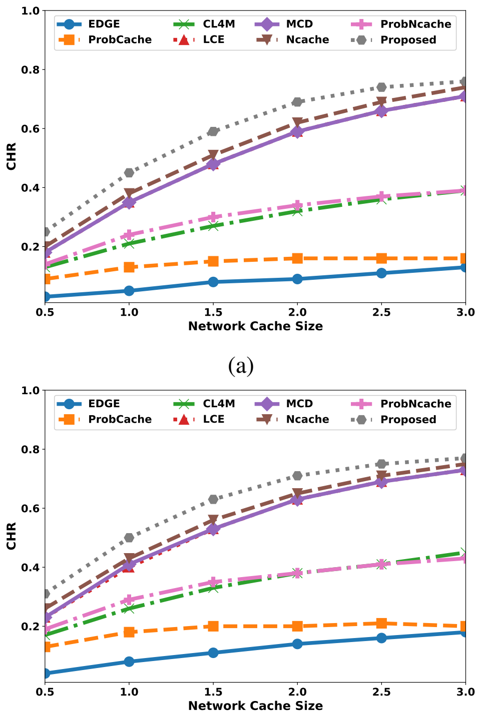
We conducted a comparative analysis of our proposed strategy and state-of-the-art caching strategies. Specifically, we compared our approach with EDGE [12], Probcache [15], LCE [8], MCD [8], CL4M [22], Ncache [11], and ProbNcache [11] on the basis of two metrics - CHR and content access latency. CHR measures the likelihood of a content request being fulfilled by an intermediate router within the network. Content access latency measures the entire duration from when a user initiates a request until it receives the content as a response.

Figure 1 shows that the proposed strategy consistently achieved a higher CHR than state-of-the-art methods across the complete range of C and α. This is because our proposed approach strategically selects the highest degree router for placing the content, ensuring widespread content accessibility across the network. Similarly, figure 2 shows that our proposed strategy achieves less content access latency as compared to state-of-the-art methods. This is because the proposed strategy needs to check only the router responsible for caching the content. Additionally, the content is strategically positioned on the highest degree router which makes the content highly accessible in the network.

C. Sensitivity Analysis

1) Varying Cache Size (C): The cache size C plays an important role in improving the overall cache performance. If C increases, the overall in-network capacity increases due to which more content can be cached. As a result, higher C leads to better CHR. This is reflected in our experiments also, as shown in Figure 3a. The CHR for our proposed strategy increased as we increased the value of C from 0.5 to 3.0 percent. Moreover, the content access latency is also reduced by increasing the cache size as shown in Figure 3b. This is because expanding the storage capacity within the network nodes offers the potential to accommodate additional content without the need for content replacement. However, increasing the cache storage also leads to an increase in the cost of the router. Thus, it is essential to choose the cache size optimally so as to achieve maximum CHR and minimum latency without any significant increase in the cost of the caching routers.

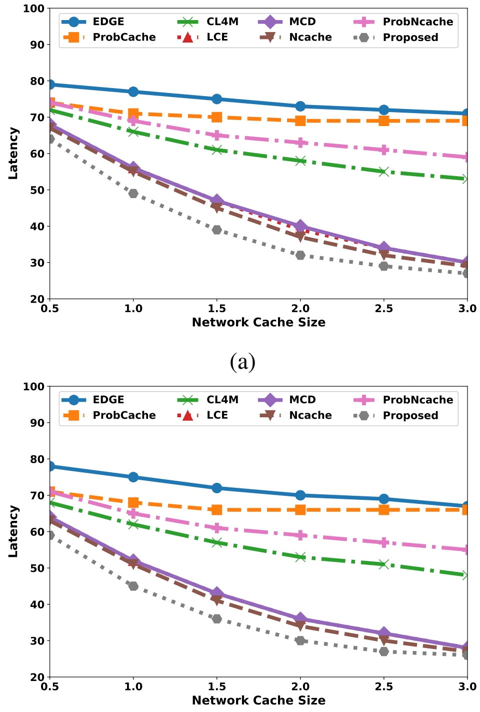
2) Varying Content Distribution Parameter (α): The content distribution parameter, α, indicates that some contents are



(a)

(b)

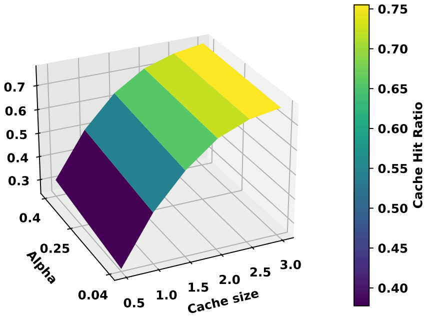
Fig. 1: CHR for (a) α = 0.04, and (b) α = 0.4



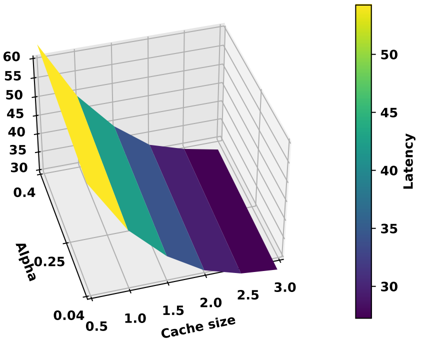
(a)

(b)

Fig. 2: Latency for (a) α = 0.04, and (b) α = 0.4



(a) CHR



(b) Latency

Fig. 3: CHR and Latency for α = [0.04 - 0.4] and cache size = [0.5 - 3.0]

more popular or frequently requested than others. This distribution indicates that a small number of content is requested more frequently than others, which leads to an increase in the popularity of those contents. As a result, the chance of the presence of those contents in a router's cache increases slightly. Thus, an increase in the value of α results in a marginal increase in the CHR. This can be noticed in Figure 3a. Moreover, due to a higher α value, a consumer can retrieve the content from an intermediate router itself instead of retrieving it from the producer. This results in a marginal lower content access latency as shown in Figure 3b.

VI. CONCLUSION

NDN is a next-generation Internet architecture proposed to replace the current TCP/IP Internet architecture due to its various shortcomings. One advantage of NDN over current Internet architecture is its ability for in-network caching. However, the known content retrieval and placement techniques suffer from various drawbacks such as high lookup repetition overhead, poor cache utilization, and high content redundancy. Through this paper, we made an attempt to improve the overall performance of NDN architecture by proposing an efficient content retrieval and content placement approach. Our approach not only strategically selects the routers for content retrieval, but also optimally decides to place the content in an appropriate router. Our experiments showed that the proposed approach could achieve better CHR and less content access latency as compared to the previously known state-of-the-art approaches.

REFERENCES

[1] Ahlgren, Bengt, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman. "A survey of information-centric networking." IEEE Communications Magazine 50, no. 7 (2012): 26-36.

[2] Benmoussa, Ahmed, Chaker Abdelaziz Kerrache, Nasreddine Lagraa, Spyridon Mastorakis, Abderrahmane Lakas, and Abdou El Karim Tahari. "Interest flooding attacks in named data networking: Survey of Existing Solutions, Open Issues, Requirements, and Future Directions." ACM Computing Surveys 55, no. 7 (2022): 1-37.

[3] An, Ying, and Xi Luo. "An in-network caching scheme based on energy efficiency for content-centric networks." IEEE Access 6 (2018): 20184-20194.

[4] Afanasyev, Alexander, Xiaoke Jiang, Yingdi Yu, Jiewen Tan, Yumin Xia, Allison Mankin, and Lixia Zhang. "NDNS: A DNS-like name service for NDN." In 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1-9. IEEE, 2017.

[5] Cao, Jianxun, Dan Pei, Xiaoping Zhang, Beichuan Zhang, and Youjian Zhao. "Fetching popular data from the nearest replica in NDN." In 2016 25th International Conference on Computer Communication and Networks (ICCCN), pp. 1-9. IEEE, 2016.

[6] Xylomenos, George, Christopher N. Ververidis, Vasilios A. Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V. Katsaros, and George C. Polyzos. "A survey of information-centric networking research." IEEE communications surveys & tutorials 16, no. 2 (2013): 1024-1049.

[7] Jacobson, Van, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. "Networking named content." In Proceedings of the 5th international conference on Emerging networking experiments and technologies, pp. 1-12. ACM, 2009.

[8] Laoutaris, Nikolaos, Sofia Syntila, and Ioannis Stavrakakis. "Meta algorithms for hierarchical web caches." In IEEE International Conference on Performance, Computing, and Communications, 2004, pp. 445-452. IEEE, 2004.

[9] Iqbal, Shahid Md Asif. "Cache-MAB: A reinforcement learning-based hybrid caching scheme in named data networks." Future Generation Computer Systems 147 (2023): 163-178.

[10] Saino, Lorenzo, Ioannis Psaras, and George Pavlou. "Icarus: a caching simulator for information centric networking (icn)." In SimuTools, vol. 7, pp. 66-75. ICST, 2014.

[11] Chaudhary, Pankaj, Neminath Hubballi, and Sameer G. Kulkarni. "NCache: neighborhood cooperative caching in named data networking." In 2022 5th International Conference on Hot Information-Centric Networking (HotICN), pp. 36-41. IEEE, 2022.

[12] Laoutaris, Nikolaos, Hao Che, and Ioannis Stavrakakis. "The LCD interconnection of LRU caches and its analysis." Performance Evaluation 63, no. 7 (2006): 609-634.

[13] Reshadinezhad, Amir, Mohammad Reza Khayyambashi, and Naser Movahedinia. "An efficient adaptive cache management scheme for named data networks." Future Generation Computer Systems 148 (2023): 79-92.

[14] Naeem, M. A., and H. Suhaidi. "A survey on probabilistic caching mechanisms in content centric networking." J. Adv. Res. Dyn. Control Syst 10, no. 10 (2018): 1309-1321.

[15] Naeem, Muhammad Ali, Shahrudin Awang Nor, Suhaidi Hassan, and Byung-Seo Kim. "Performances of probabilistic caching strategies in content centric networking." IEEE access 6 (2018): 58807-58825.

[16] Psaras, Ioannis, Wei Koong Chai, and George Pavlou. "Probabilistic in-network caching for information-centric networks." In Proceedings of the second edition of the ICN workshop on Information-centric networking, pp. 55-60. ACM, 2012.

[17] Ioannou, Andriana, and Stefan Weber. "A survey of caching policies and forwarding mechanisms in information-centric networking." IEEE Communications Surveys & Tutorials 18, no. 4 (2016): 2847-2886.

[18] Wu, Haibo, Jun Li, Jiang Zhi, Yongmao Ren, and Lingling Li. "A hybrid ICN caching strategy based on region division." In Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies, pp. 78-79.ACM, 2019.

[19] Wang, Yu, Mingwei Xu, and Zhen Feng. "Hop-based probabilistic caching for information-centric networks." In 2013 IEEE Global Communications Conference (GLOBECOM), pp. 2102-2107. IEEE, 2013.

[20] Meng, Yahui, Muhammad Ali Naeem, Rashid Ali, and Byung-Seo Kim. "EHCP: An efficient hybrid content placement strategy in named data network caching." IEEE Access 7 (2019): 155601-155611.

[21] Naeem, Muhammad Ali, Tu N. Nguyen, Rashid Ali, Korhan Cengiz, Yahui Meng, and Tahir Khurshaid. "Hybrid cache management in IoT-based named data networking." IEEE Internet of Things Journal 9, no. 10 (2021): 7140-7150.

[22] Chai, Wei Koong, Diliang He, Ioannis Psaras, and George Pavlou. "Cache "less for more" in information-centric networks." In NETWORKING 2012: 11th International IFIP TC 6 Networking Conference, Prague, Czech Republic, May 21-25, 2012, Proceedings, Part I 11, pp. 27-40. Springer Berlin Heidelberg, 2012.

[23] Lal, Kumari Nidhi, and Anoj Kumar. "A centrality-measures based caching scheme for content-centric networking (CCN)." Multimedia Tools and Applications 77 (2018): 17625-17642.

[24] Naeem, Muhammad Ali, Muhammad Atif Ur Rehman, Rehmat Ullah, and Byung-Seo Kim. "A comparative performance analysis of popularity-based caching strategies in named data networking." IEEE Access 8 (2020): 50057-50077.

[25] Ren, Jing, Wen Qi, Cedric Westphal, Jianping Wang, Kejie Lu, Shucheng Liu, and Sheng Wang. "Magic: A distributed max-gain in-network caching strategy in information-centric networks." In 2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 470-475. IEEE, 2014.

[26] Yu, Meiju, and Ru Li. "Dynamic popularity-based caching permission strategy for named data networking." In 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), pp. 576-581. IEEE, 2018.

[27] Kumar, Sumit, and Rajeev Tiwari. "Optimized content centric networking for future internet: dynamic popularity window based caching scheme." Computer Networks 179 (2020): 107434.

[28] Zhang, Ran, Jiang Liu, Renchao Xie, Tao Huang, F. Richard Yu, and Yunjie Liu. "Service-aware optimal caching placement for named data networking." Computer Networks 174 (2020): 107193.

[29] Naeem, Muhammad Ali, Shahrudin Awang Nor, Suhaidi Hassan, and Byung-Seo Kim. "Performances of probabilistic caching strategies in content centric networking." IEEE access 6 (2018): 58807-58825.

[30] Hou, Rui, Lang Zhang, Tingting Wu, Tengyue Mao, and Jiangtao Luo. "Bloom-filter-based request node collaboration caching for named data networking." Cluster Computing 22 (2019): 6681-6692.

[31] Kalghoum, Anwar, and Leila Azouz Saidane. "Fcr-ns: a novel caching and forwarding strategy for named data networking based on software defined networking." Cluster Computing 22 (2019): 981-994.

[32] Herouala, Abdelkader Tayeb, Benameur Ziani, Chaker Abdelaziz Kerrache, Abdou el Karim Tahari, Nasreddine Lagraa, and Spyridon Mastorakis. "CaDaCa: a new caching strategy in NDN using data categorization." Multimedia Systems (2022): 1-16.

[33] Amadeo, Marica, Giuseppe Ruggeri, Claudia Campolo, Antonella Molinaro, and Giuseppe Mangiullo. "Caching popular and fresh IoT contents at the edge via named data networking." In IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 610-615. IEEE, 2020.

[34] Bernardini, César, Thomas Silverston, and Olivier Festor. "MPC: Popularity-based caching strategy for content centric networks." In 2013 IEEE international conference on communications (ICC), pp. 3619-3623. IEEE, 2013.

[35] Ming, Zhongxing, Mingwei Xu, and Dan Wang. "Age-based cooperative caching in information-centric networking." In 2014 23rd International Conference on Computer Communication and Networks (ICCCN), pp. 1-8. IEEE, 2014.

[36] Naeem, Muhammad Ali, Shahrudin Awang Nor, Suhaidi Hassan, and Byung-Seo Kim. "Compound popular content caching strategy in named data networking." Electronics 8, no. 7 (2019): 771.

[37] Lee, Sung-Won, Dabin Kim, Young-Bae Ko, Jae-Hoon Kim, and Myeong-Wuk Jang. "Cache capacity-aware CCN: Selective caching and cache-aware routing." In 2013 IEEE Global Communications Conference (GLOBECOM), pp. 2114-2119. IEEE, 2013.

[38] Ong, Mau Dung, Min Chen, Tarik Taleb, Xiaofei Wang, and Victor CM Leung. "FGPC: Fine-grained popularity-based caching design for content centric networking." In Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems, pp. 295-302.ACM, 2014.

[39] Topology-Zoo, http://www.topology-zoo.org/dataset.html, (accessed on 31-10-2023).