Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 18 No. 16 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijim.v18i16.49185

PAPER

LPCE-Based Replacement Scheme for Enhancing Caching Performance in Named Data Networking

Samir Nassane[1](✉), Sid Ahmed Mokhtar Mostefaoui[2], Bendaoud Mebarek[2], Abdelkader Alem[1]

[1]Laboratoire de Génie Energétique et Génie Informatique (L2GEGI), University of Tiaret, Tiaret, Algeria

[2]Laboratoire de Recherche en Intelligence Artificielle et Systèmes (LRIAS), University of Tiaret, Tiaret, Algeria

samir.nassane@univ-tiaret.dz

ABSTRACT

The substantial surge in users has adversely impacted the performance of the present IP-based Internet. Named data networking (NDN) emerges as a future alternative, given its distributed content caching system, where data can be cached in multiple routers and retrieved from the closest one instead of the original producer, enhancing content availability, reducing latency, and minimizing data loss. This paper introduces the less popular content eviction (LPCE) policy, a novel cache replacement scheme designed to enhance the caching performance of the conventional LFU (least frequently used) policy in NDN routers, thereby improving overall network efficiency. The proposed method subsumes LFU and FIFO (first in first out) policies and employs an additional list controlled by the LRU (least recently used) policy. Utilizing the ccnSim simulator, we conduct a comparison of LPCE's performance with that of the LFU technique and other competing caching techniques, considering variations in several simulation parameters. Experimental results reveal that the proposed LPCE algorithm excels over others across a majority of performance metrics, such as cache hit ratio, content delivery delay, upstream hop count, network traffic, and producers' load. Besides, the findings indicate that LPCE outperforms LFU, with an increase in cache hit ratio ranging from 1.32% to 5.75%.

KEYWORDS

named data networking (NDN), distributed content caching, replacement policies, least frequently used (LFU) policy, least recently used (LRU) policy, less popular content eviction (LPCE) policy, cache hit

1 INTRODUCTION

The current Internet has seen significant advancements and has become an essential aspect of contemporary existence. From being a simple IP network, it has evolved into a global platform offering a wide range of online services, including social networks, online banking, and video streaming. Internet users now have additional needs, including the requirement for high bandwidth, low latency, mobility, data availability, and security, as a result of the advent of these new services.

Nassane, S., Mostefaoui, S.A.M., Mebarek, B., Alem, A. (2024). LPCE-Based Replacement Scheme for Enhancing Caching Performance in Named Data Networking. *International Journal of Interactive Mobile Technologies (iJIM)*, 18(16), pp. 119–141. https://doi.org/10.3991/ijim.v18i16.49185

Article submitted 2024-03-18. Revision uploaded 2024-05-26. Final acceptance 2024-06-09.

These challenges are simply a small sample of the myriad issues that the Internet faces today [1]. To address this situation, a novel network architecture is suggested to supplement or potentially replace the existing Internet architecture; it is known as named data networking (NDN) [2, 3]. NDN networks are among the proposed architectural approaches for the basic ICN (information-centric networking) model [4]. Indeed, NDN differs from an IP-based Internet in that it considerably boosts network performance with its exceptional caching ability and innovative content naming scheme [5]. For example, Figure 1 illustrates the data retrieval model on the present Internet, wherein client requests travel through the Internet backbone, which is without any caching process, to reach the data source. This approach causes heightened delays and losses in data delivery, overloads the resources of the traversed routers, accelerates network congestion, and further saturates central servers. These factors negatively influence Internet performance, especially as the number of users continues to increase [5].

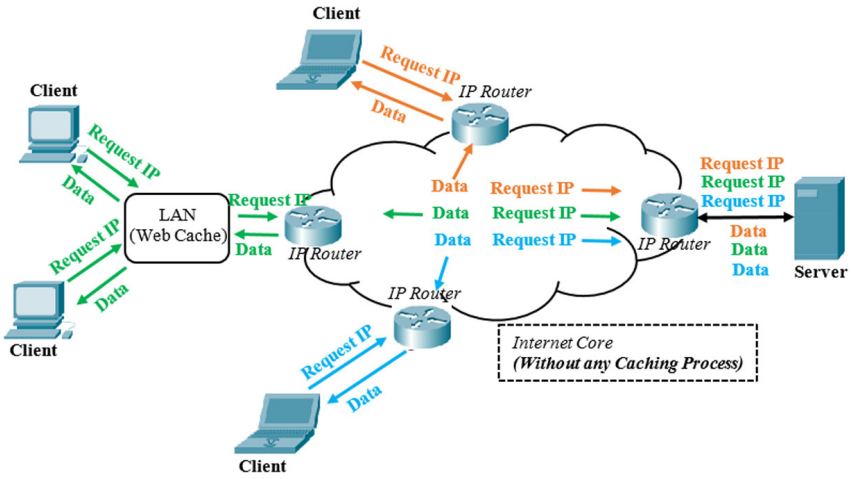

Fig. 1. The data retrieval model on the present Internet

In contrast, inside the NDN framework, as shown in Figure 2, the caching procedure is distributed across all routers. This implies that data can be stored in multiple routers and retrieved from the closest one, enhancing the efficiency of content distribution and markedly alleviating the load on central producers. Thus, the NDN caching strategy significantly improves content availability, reduces latency, minimizes data loss, and decreases network traffic, as consumers demand data from the nearest router rather than directly from the original producer [6, 7].

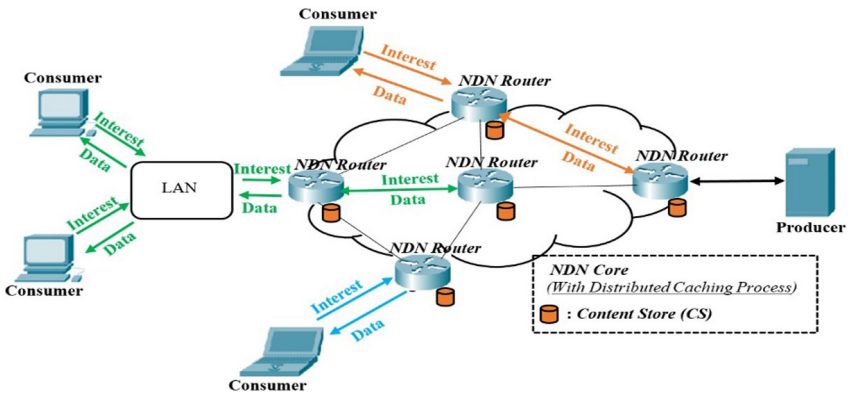

Fig. 2. The model for retrieving data within the NDN architecture

To implement this caching process easily, NDN routers opt for content names as network addresses instead of IP addresses. This choice is made because these names are unique and independent of the servers' locations. Simply put, to request data on the NDN network, the consumer must indicate the content name rather than supplying an IP location address [8]. Afterward, the network searches for the name of this content on the nearest routers and sends back the associated data (see Figure 2).

As depicted in Figure 3, the NDN router consists of three primary components: the pending interest table (PIT), which serves as a list of unsatisfied interests; the forwarding information base (FIB), which is employed for routing interest packets; and the content store (CS), which is utilized to store copies of frequently requested data [9].

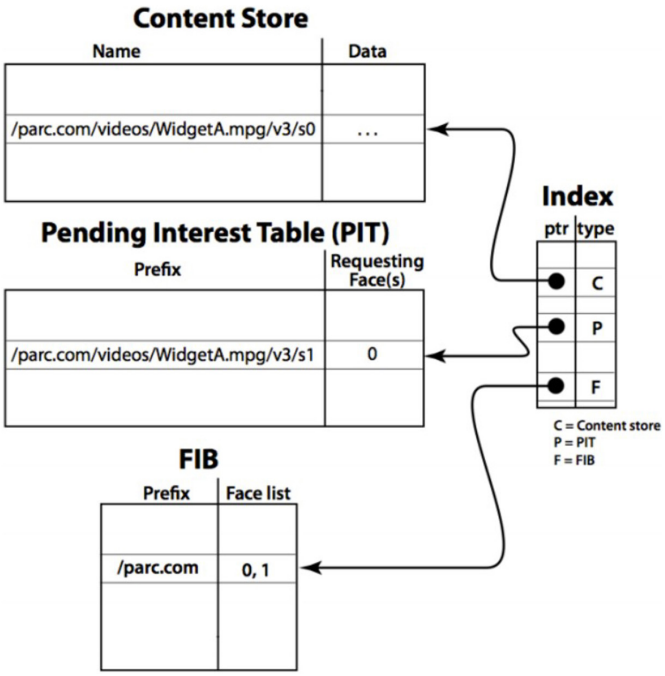

Fig. 3. The caching components of an NDN router [9]

When an interest in data is identified, the router verifies whether the solicited data is in its cache; if it is, it can be promptly retrieved. The steps involved in how an NDN router handles an interest packet are illustrated in Figure 4 and detailed below [10]:

Initially, the router checks for the presence of the related data in its CS cache. If found, it transmits the data via the incoming interface of the received interest. Otherwise, the router inspects the PIT table. In the event that there is already an interest in this data, the router simply appends the incoming interface of the interest to the corresponding entry. If there is no existing interest, the router generates a new interest entry in the PIT and then moves on to the FIB base to locate the interfaces leading to the demanded data. Afterward, interest packets will be sent through all the associated output links.

In Figure 4, the process by which an NDN router deals with a data packet is elaborated as follows [10]: At first, the existence of the corresponding interest in the PIT table is verified. If not found, the data is rejected. However, if an entry is identified, the data is subsequently transmitted through all the interfaces linked to this interest. Subsequently, the data is cached in CS, and the related interest is discarded from the PIT table. In this process, it should be noted that no role is played by the FIB table.

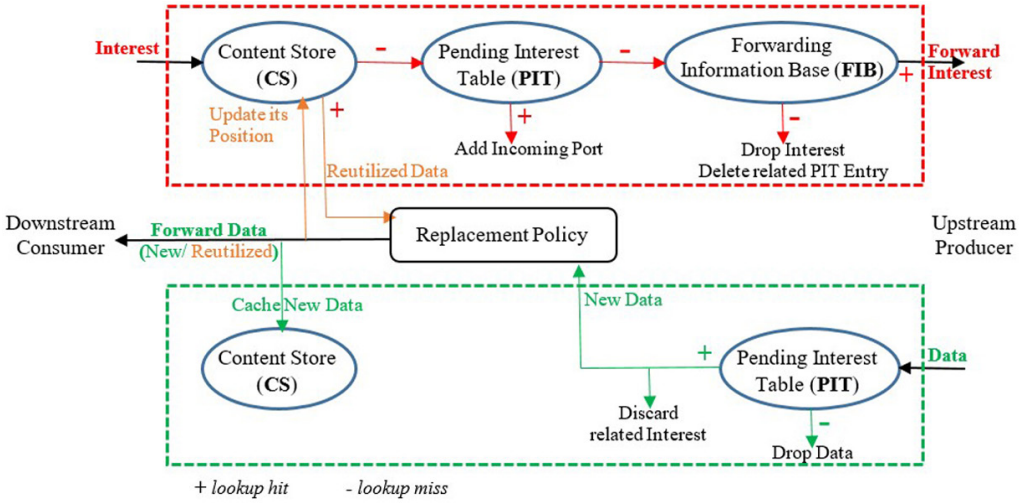

Fig. 4. The NDN router's processing for Interest and data packets

The content caching system stands as a fundamental pillar of the NDN architecture, providing an effective remedy for the rising traffic by storing named data duplicates within the routers along the path and guaranteeing swift access when required. It comprises two essential parts, namely the cache placement (or decision) strategy and the cache replacement (or eviction) policy [11].

The goal of the placement strategy is to identify the router where the content should be saved [12]. As examples, we refer to the LCE (leave copy everywhere) and LCD (leave a copy down) strategies [13]. In the context of LCE, each NDN router along the path to the inquiring consumer stocks the desired content [14]. With LCD, the routers downstream exclusively duplicate the required data [15]. On the other hand, due to the constrained size of the CS, the replacement policy is intended to pick content for eviction from the cache, thereby making room for new ones [16], [17].

The performance of the caching system is a crucial factor that can significantly impact the overall performance of NDN networks. Consequently, it has attracted considerable attention from the networking research community. Thus, several caching policies have been proposed or enhanced to improve the network's overall performance. Caching performance is typically measured using various metrics such as cache hit ratio, producer load, network traffic, and latency. The cache hit rate is the most important metric for measuring performance in NDN. A higher cache hit rate implies that more consumer requests are satisfied from network caches, resulting in lower latency, reduced network traffic, and less strain on content producers.

In this paper, we propose the less popular content eviction (LPCE) policy, a new cache replacement scheme aimed at enhancing the caching performance of the least frequently used (LFU) replacement policy in NDN routers, thereby improving the entire network's performance. The subsequent sections of this paper are structured as follows: Related work is explored in Section 2; our novel caching policy is introduced in Section 3; simulation parameters and scenarios are described in Section 4; the simulation results are presented in Section 5; and finally, a summary and future work are provided in Section 6.

2 RELATED WORK

Caching performance in NDN networks relies on two key factors: content placement strategy and content replacement policy. In this context, we highlight

prominent replacement policies that align with our approach and have been proposed or improved to enhance the whole network's performance. These replacement policies can be categorized into three types: those relying on the popularity (or frequency) of data, those dependent on the recency of data, and those considering both the popularity and recency of data [10, 14].

In [18], the LFU policy was proposed for NDN networks. This policy is predicated on the assumption that frequently demanded content might be asked for again soon. As a result, the strategy involves clearing the content with the lowest popularity from the cache to create space for new arrivals. However, content with a high popularity could endure within the CS even when not actively requested, leading to an issue referred to as the 'aging phenomenon' or 'cache pollution.' Several LFU variants, including LFU-aging, LFU with dynamic aging, and Window-LFU, have been proposed as solutions to address this issue and thereby enhance the entire network's performance. In fact, our approach follows this direction. It aims to enhance LFU performance by reducing cache pollution.

As presented in [19], LFU-Aging minimizes cache pollution by halving the popularity of each cached content whenever the average popularity surpasses a predefined threshold. In the same context, authors in [20] introduced LFU with dynamic aging (LFUda) as an enhancement to LFU-Aging to mitigate the risk of cache pollution.

As proposed in [21] for NDN networks, the least recently/frequently used (LRFU) policy combines the LRU (least recently used) and LFU policies. At full capacity, LRFU selects the data with the smallest CRF (combined recency frequency) value for eviction. The experimental results indicated that the LRFU replacement policy achieved a 3.36% higher hit rate than the LRU and a 5.78% higher hit rate compared with the priority-FIFO replacement policy.

The window-LFU (WLFU) policy was originally implemented for web caching in [22]. Here, the LFU method receives enhancement through the integration of a sliding window mechanism. Notably, window-LFU exhibits enhanced adaptability to dynamic changes in content distribution, outperforming LFU by concentrating solely on this specific temporal window.

The name popularity algorithm (NPA) [23] intends to boost the performance of the LFU replacement policy in NDN networks by employing an additional table, HT, to record the recent popularity of multiple accessed contents. Hence, when previously dislodged popular data from the CS cache is revisited, it regains its historical popularity (if available) from the HT table, thereby enhancing cache efficiency. The simulation findings reveal that the NPA algorithm excels in comparison to the LFU.

The two-queue (2Q) replacement policy was proposed in [24]. This caching technique aims to improve the LRU policy by adhering to the principle that when a content genuinely requires caching, it ought to consistently receive periodic requests, especially following a significant number of accesses within a short time frame. As a result, the 2Q policy manages a FIFO queue and two LRU lists. In practical applications, the 2Q algorithm is more effective than the LRU algorithm.

The segmented-LRU (SLRU) method [25] is designed to avoid the premature disposal of temporally popular data. Hence, the cache is split into two different LRU lists: A1 for probationary data and A2 for protected data. New contents are continually directed to the probation list. If content from the A1 list is revisited, it gets transferred to the A2 list. In this manner, data with temporary popularity, upon entering the protected list, will have an extended duration in the cache compared to less popular data. Findings display that the SLRU achieves a 9% and 12% higher cache hit rate compared to the LRU and FIFO replacement policies, respectively.

The adaptive replacement cache (ARC) policy [26] enables the boost of LRU performance through the management of two LRU lists with variable sizes: T1 for data retrieved once and T2 for data retrieved on a minimum of two occasions. In addition, ARC stores only the names, not the content, of recently evicted data from both lists. It utilizes two additional LRU lists: B1 for managing content newly removed from the T1 cache and B2 for handling content recently removed from the T2 cache. Results show that ARC achieves a 4% higher hit rate than the LRU replacement policy.

An analysis of the previously mentioned studies reveals that replacement approaches yielding favorable caching performance outcomes often incorporate two key concepts: ghost entries and cache partitioning.

Ghost entries are exclusively dedicated to safeguarding metadata (names or keys) that identify contents recently forced out. To clarify, when content is taken out of a cache, only its name remains stored in a ghost list. Owing to their low memory consumption, several replacement policies keep a considerable quantity of ghost entries. This incurs additional costs on the one hand, but it also enhances the efficiency and performance of the cache on the other.

Regarding the second concept, the cache is commonly split into two segments. This is done to improve caching performance either by combining two different replacement policies (such as ARC) or by merging two distinct ideas while using the same caching technique (such as SLRU).

Our approach, detailed in the following section, aligns perfectly with this context. In simpler terms, it is designed based on the two aforementioned notions, aiming to further enhance the caching performance of the classical LFU scheme and, consequently, improve the overall network performance.

3 PROPOSED METHODOLOGY

As noted, before the proposed LPCE algorithm aims to improve the performance of the typical LFU method. As a reminder, the LFU strategy removes the least popular content from the cache to make room for new data. However, highly popular content can persist in the cache even when it is not frequently requested, causing an issue known as the “aging phenomenon” or “cache pollution.” Therefore, our LPCE method is designed to reduce this cache pollution, thereby enhancing network caching performance. This is achieved by partitioning the LFU cache into two subcaches with fixed capacities: one designated as the secondary cache, represented by the FIFO queue, and the other as the main cache, denoted by the LFU table. In addition, the FIFO cache occupies 5% of the overall cache space, with the main cache accounting for the remaining 95%.

A supplemental LRU ghost list is also incorporated to sustain the popularity of recently expelled data for an extended period of time. Properly speaking, when data is excluded from the cache, its popularity endures, guaranteeing that the ejected contents preserve their popularity and consequently aid in refining the caching decision in the near future.

In practice, our proposed algorithm, LPCE, combines LFU, LRU, and FIFO policies while leveraging an important number of ghost entries to boost cache efficiency. It is important to note that all these strategies ensure the ordered structure of their lists. The general architecture of our proposed LPCE policy is depicted in Figure 5.

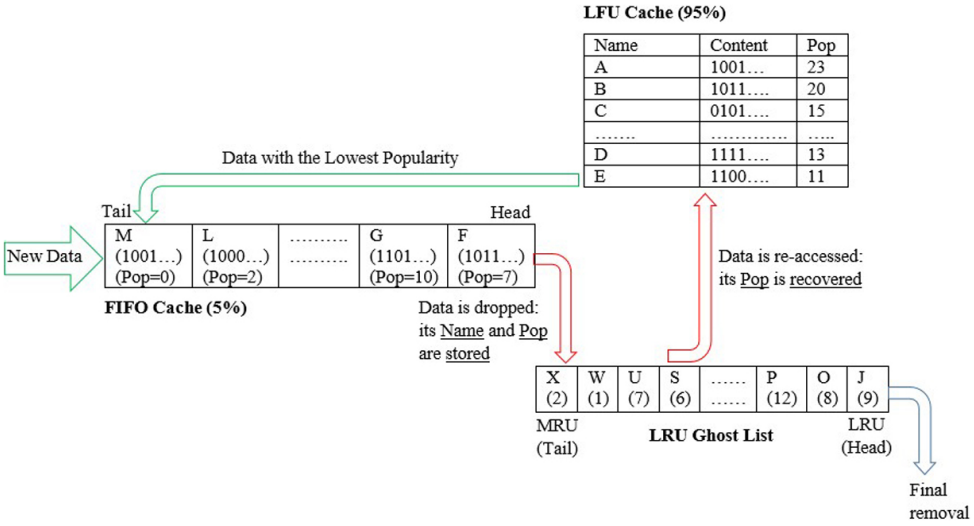

Fig. 5. The general design of the proposed LPCE policy

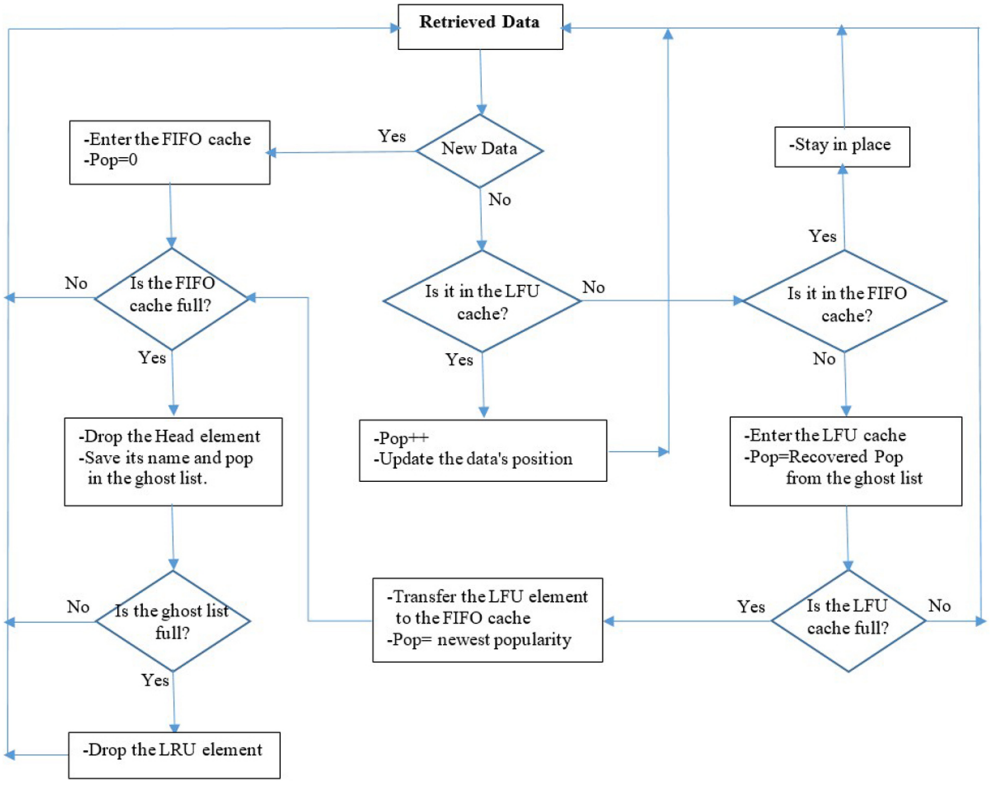

Fig. 6. The flowchart of the proposed LPCE algorithm

As displayed in Figure 6, this is how the proposed LPCE algorithm operates:

1. Upon receiving new data, it first enters the FIFO secondary cache and stays in place as long as it is actively consumed, thereby ensuring the processing of its initial requests within the FIFO cache. This helps absorb a burst of data and weaken the aging process to the fullest extent achievable.
2. After reaching a certain age, it is eliminated in accordance with the FIFO policy, but its name and popularity are kept on the ghost list.

3. If content that has been recently dropped from the FIFO cache is reutilized and its name still appears among the ghost entries, it is then incorporated into the LFU cache with its restored popularity from the ghost list. This indicates the popularity of the content, suggesting a high likelihood of future interests and, consequently, justifying its caching. It is crucial to underline that recovering the previous popularity enhances the ranking of data in the LFU cache, safeguarding it against any premature rejection.
4. Otherwise, its name and popularity will be conclusively deleted from the ghost list under the LRU policy, suggesting it is either unpopular or old content. Through this method, our algorithm discards aged content that could potentially pollute the LFU cache.
5. After attaining complete capacity, the primary LFU cache transfers the less popular content to the secondary cache along with its newest popularity. This process affords the LFU element a fresh chance for use before any probable elimination. Consequently, data with moderate popularity, once entering the main cache, will enjoy a longer lifespan in the cache compared to both unpopular and elderly data.

Note that the popularity of a data point is only incremented within the main LFU cache.

4 SIMULATION PARAMETERS AND SCENARIOS

In this section, we detail the simulation setup used to implement and measure the performance of our LPCE approach. We use ccnSim [27] for the simulation. The ccnSim is an open-source simulator with high scalability, developed using C++ to extend the OMNET++ environment [28] for simulating NDN networks at the data chunk level.

Furthermore, we assess the LPCE algorithm's performance by comparing it with five different policies (LFU, LFUda, WLFU, NPA, and 2Q) under identical experimental conditions. This evaluation was conducted after we extended the ccnSim simulator to include the LPCE policy and the other five policies. The performance evaluation metrics used in the simulation are explained in the next section. The cache hit rate is a crucial metric for evaluating performance in NDN. A higher cache hit rate indicates that more consumer interests are served by routers, leading to lower content delivery delays, decreased network load, and less burden on producers. Table 1 presents the main simulation parameters.

We use two network topologies (TREE and CDN), predefined in ccnSim, each differing in size as well as the number of consumers and producers (see Figures 7 and 8). The TREE topology contains 15 routers, eight consumers, and one producer. However, the CDN topology contains 67 routers, 32 consumers, and five producers. The producer (data source) provides a catalog of 10^6 different contents for consumers. The content store (CS) is configured by default with 1000 data chunks in each router. The consumer (data requester) sends 20 interest packets per second (default value) with a randomized time gap between two consecutive interests. The contents requested in interest packets follow the Mandelbrodt-Zipf [29] probability distribution used as a popularity model with a default α parameter value of 1.

In this simulation, we did not incorporate any routing protocols. Instead, the simulation employs a forwarding strategy known as SPR (shortest path routing), which establishes in each router a FIB table to forward interest packets towards the producer via the shortest path. This path is determined by the simulator using the Dijkstra algorithm [27]. The FIB table functions by mapping name-prefix-based interest packets to the output interfaces that lead to the requested data (see Figure 3).

Over a long simulation duration, we study various scenarios involving changes in diverse simulation parameters such as placement strategy (PS), cache size (CS), consumer interest rate (Int_rate), MZipf α, and topology type (T) to evaluate the caching performance of the proposed LPCE under different conditions. Table 2 lists the simulation scenarios applied in this study.

To carry out this simulation, we need a machine equipped with robust hardware capabilities. Therefore, we employ an HP Zbook Fury G8 mobile workstation PC, featuring an Intel i7-11850H CPU and 32GB RAM.

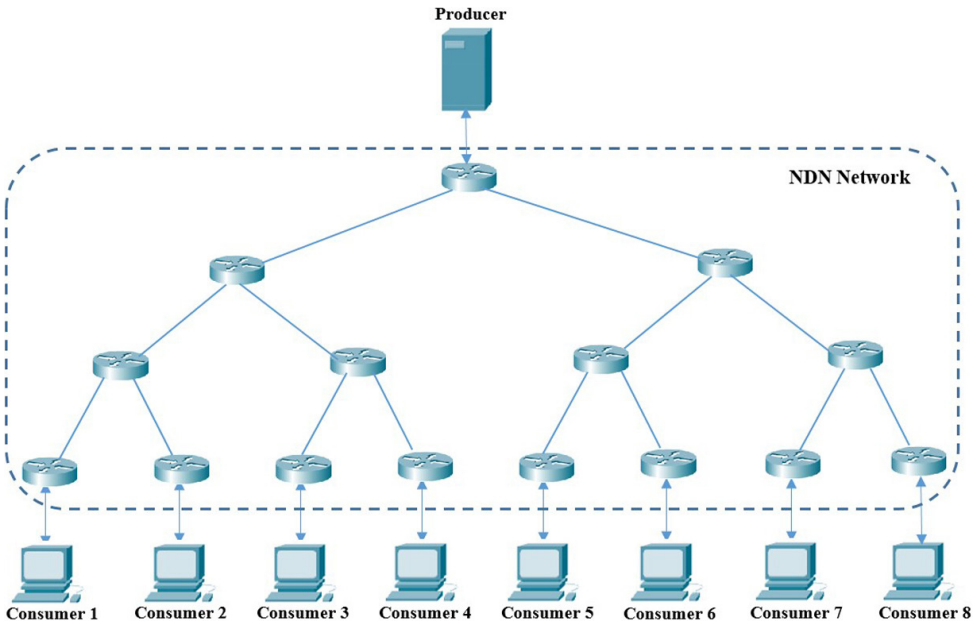

Fig. 7. The TREE topology (15 routers, 1 producer, 8 consumers)

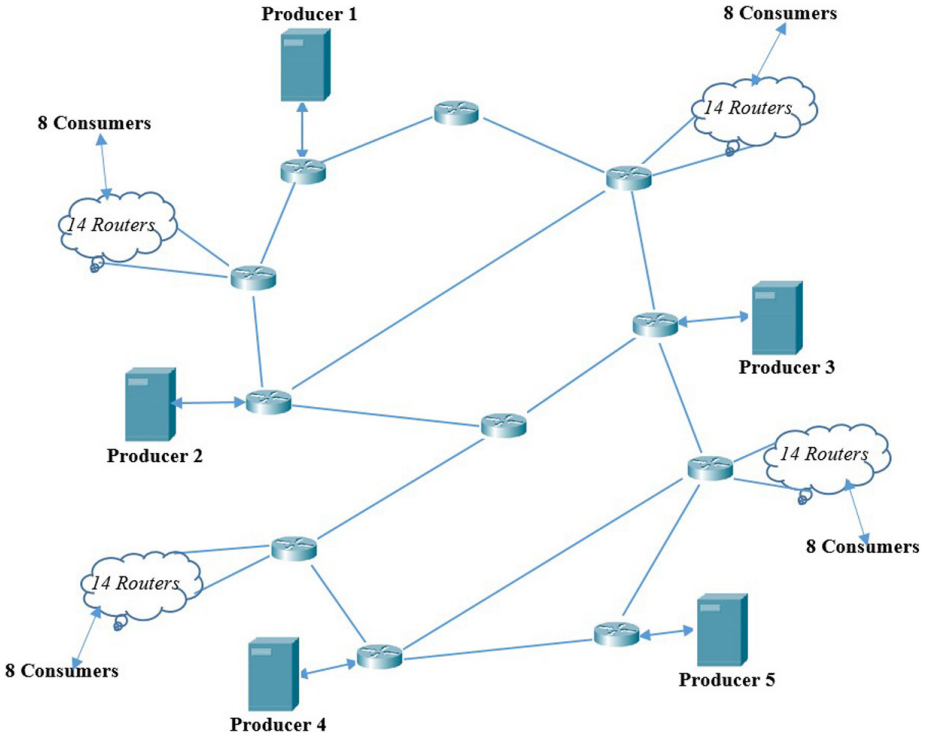

Fig. 8. The CDN topology (67 routers, 5 producers, 32 consumers)

Table 1. Simulation parameters [21, 30, 31]

Parameter	Values
Topology (T)	TREE, CDN
Number of routers (N)	15 (TREE), 67 (CDN)
Number of producers	1 (TREE), 5 (CDN)
Number of consumers	8 (TREE), 32 (CDN)
Link rate	1 Mbps
Link delay	1 ms
Contents catalog size	10^6 Diverse contents
Content size	1 Chunk
Replicas	1
Cache size (CS)	1000 Contents
Consumer interest rate (Int_rate)	20 int/s (with randomized inter-interest intervals)
MZipf α (for content popularity)	1
Placement strategy (PS)	LCE, LCD
Forwarding strategy	SPR (Shortest Path Routing)
Replacement policy	LFU, LFUda, WLFU, NPA, 2Q, LPCE
Performance metric	Cache Hit, Delay, Hop count, Network traffic
Simulation time	50000s (≈ 14 hours)

Table 2. Utilized scenarios in simulation [21, 30, 31]

No	Scenario	Variation	Values
A1	T = TREE, CS = 1000, Int_rate = 20, $\alpha = 1$	PS	LCE, LCD
A2	T = CDN, CS = 1000, Int_rate = 20, $\alpha = 1$	PS	LCE, LCD
B1	T = TREE, PS = LCE, Int_rate = 20, $\alpha = 1$	CS	1000, 2000, 4000, 5000, 10000
B2	T = CDN, PS = LCE, Int_rate = 20, $\alpha = 1$	CS	1000, 2000, 4000, 5000, 10000
C1	T = TREE, PS = LCE, CS = 1000, $\alpha = 1$	Int_rate	5, 20, 50, 100, 200, 300
C2	T = CDN, PS = LCE, CS = 1000, $\alpha = 1$	Int_rate	5, 20, 50, 100, 200, 300
D1	T = TREE, PS = LCE, Int_rate = 20, CS = 1000	MZipf α	0.8, 0.9, 1.0, 1.1, 1.2
D2	T = CDN, PS = LCE, Int_rate = 20, CS = 1000	MZipf α	0.8, 0.9, 1.0, 1.1, 1.2
E1	T = TREE, PS = LCD, Int_rate = 20, $\alpha = 1$	CS	1000, 2000, 4000, 5000, 10000
E2	T = CDN, PS = LCD, Int_rate = 20, $\alpha = 1$	CS	1000, 2000, 4000, 5000, 10000
F1	T = TREE, PS = LCD, CS = 1000, $\alpha = 1$	Int_rate	5, 20, 50, 100, 200, 300
F2	T = CDN, PS = LCD, CS = 1000, $\alpha = 1$	Int_rate	5, 20, 50, 100, 200, 300
G1	T = TREE, PS = LCD, Int_rate = 20, CS = 1000	MZipf α	0.8, 0.9, 1.0, 1.1, 1.2
G2	T = CDN, PS = LCD, Int_rate = 20, CS = 1000	MZipf α	0.8, 0.9, 1.0, 1.1, 1.2

5 RESULTS AND DISCUSSION

This work considers the following performance measures: the average cache hit rate for all caches (*C_Hit_Ratio*), the average delay in delivering content (*A_Delay*), the mean upstream hop count (*H_Count*), the total producers' load (*Pr_Load*), and the network traffic (*Net_Traffic*).

C_Hit_Ratio: It reflects the percentage of successfully satisfied interests across all routers in relation to the total number of received interests. It is determined as follows:

$$C\_Hit\_Ratio = \frac{\sum_{k=1}^{N} hits_k}{\sum_{k=1}^{N} total\_interest_k} \times 100 \quad (1)$$

Where: N is the total number of routers, $hits_k$ is the number of times a requested data in an interest packet is found in the cache of router k, and $total\_interest_k$ is the total number of interest packets received by router k.

Delay: it indicates the average time it takes for a consumer to obtain the content after submitting a related interest, computed by:

$$A\_Delay = \frac{global\ average\ delay}{number\ of\ consumers} \quad (2)$$

H_Count: the average number of upstream hops from a consumer to a router that has successfully met its interests is calculated as:

$$H\_Count = \frac{global\ average\ hops\ count}{number\ of\ consumers} \quad (3)$$

Pr_Load: It reveals the whole quantity of unfulfilled interests along the route to all producers, accompanied by the corresponding data, as determined below:

$$Pr\_Load = (interests\ received + data\ generated)\ by\ all\ producers \quad (4)$$

Net_Traffic: This corresponds to the total number of interest and data packets received across all routers, defined as:

$$Net\_Traffic = (interests + data)\ received\ by\ all\ routers \quad (5)$$

5.1 Scenarios A1 and A2

Figure 9 depicts the network performance in terms of cache hit rate, considering the two simulation scenarios, A1 and A2. It illustrates that our proposed LPCE method consistently outperforms competing policies across both topologies and placement strategies, with a higher average cache hit ratio. Under the TREE topology and LCD placement strategy, LPCE surpasses LFU with a 4.03% higher hit ratio, topping 2Q by 3.46%, LFUda by 0.98%, WLFU by 0.46%, and the NPA replacement algorithm by 0.36%.

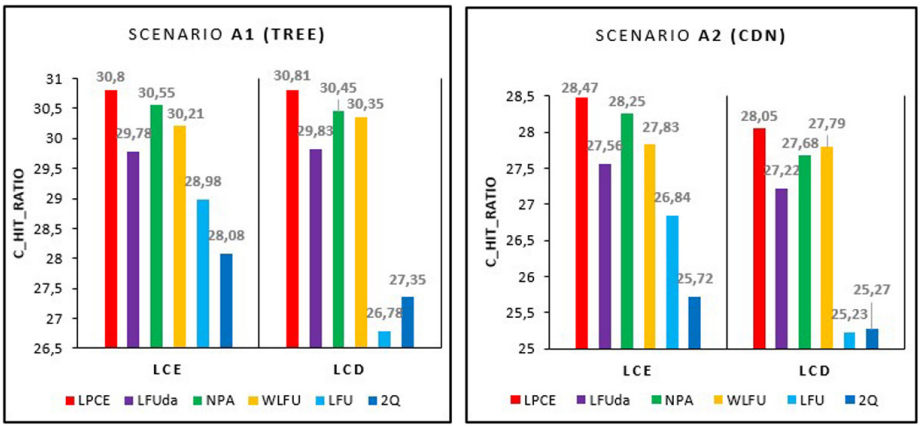

Fig. 9. Evaluation of the average cache hit ratio in scenarios A1 and A2

In the A2 scenario context, Figures 10 and 11 display the LPCE policy's performance compared to other policies. Across all the considered metrics, LPCE demonstrates superior performance. For instance, Figure 11A illustrates that LPCE generates minimal network traffic, resulting in the lowest producers' load, as shown in Figure 11B. By employing the LCE placement strategy, LPCE decreases network traffic by 9.6×10^6 packets compared to the LFU policy and by 1.6×10^6 packets compared to the NPA policy (see Figure 11A).

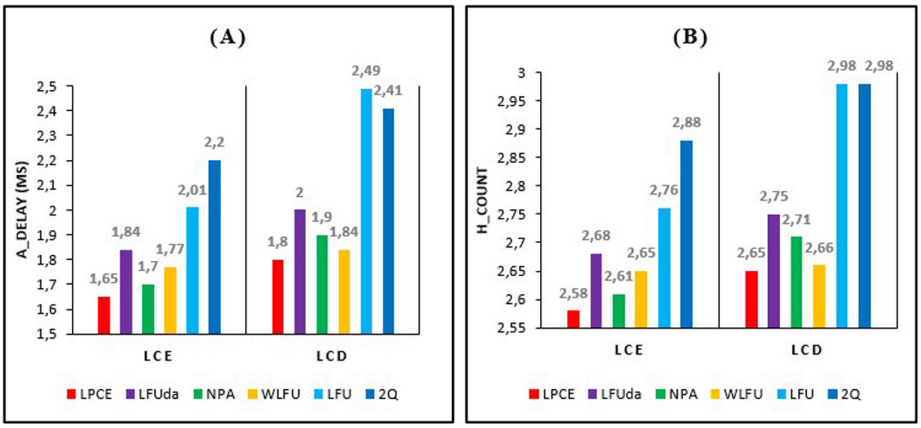

Fig. 10. Average content delivery delay and average hop count in scenarios A2

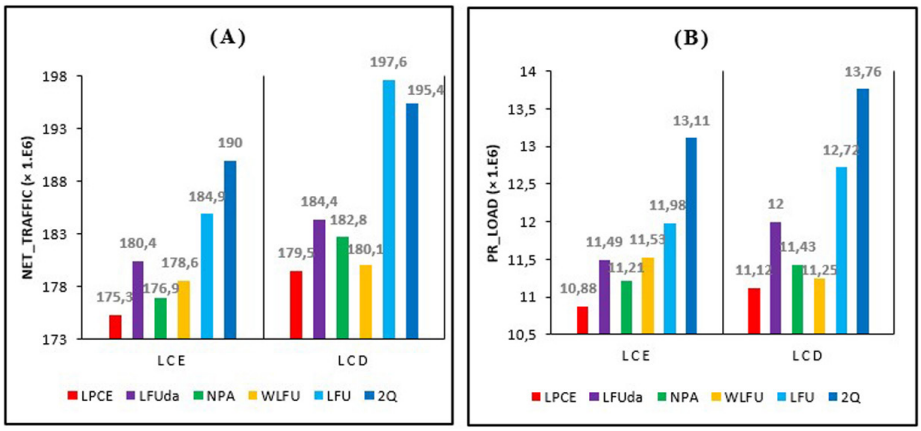

Fig. 11. Evaluation of the network traffic and the producers' load in scenario A2

5.2 Scenarios B1 and B2

Figures 12 and 13 show how well LPCE performed compared to five other caching strategies as the cache size changed. LPCE consistently outperformed the others in both scenarios (B1 and B2). As an illustration, in scenario B2 and with a cache size of 10,000 contents (see Figure 13), LPCE performs better than LFU, achieving a hit ratio that is 1.72% higher. In addition, it surpasses 2Q by 5.12%, LFUda by 1.56%, NPA by 0.46%, and the WLFU approach by 0.26%. We notice that the cache hit ratio improves for all caching policies as the size of the CS increases. This is because a larger CS increases the chances of finding the requested content along the route before reaching the content producer.

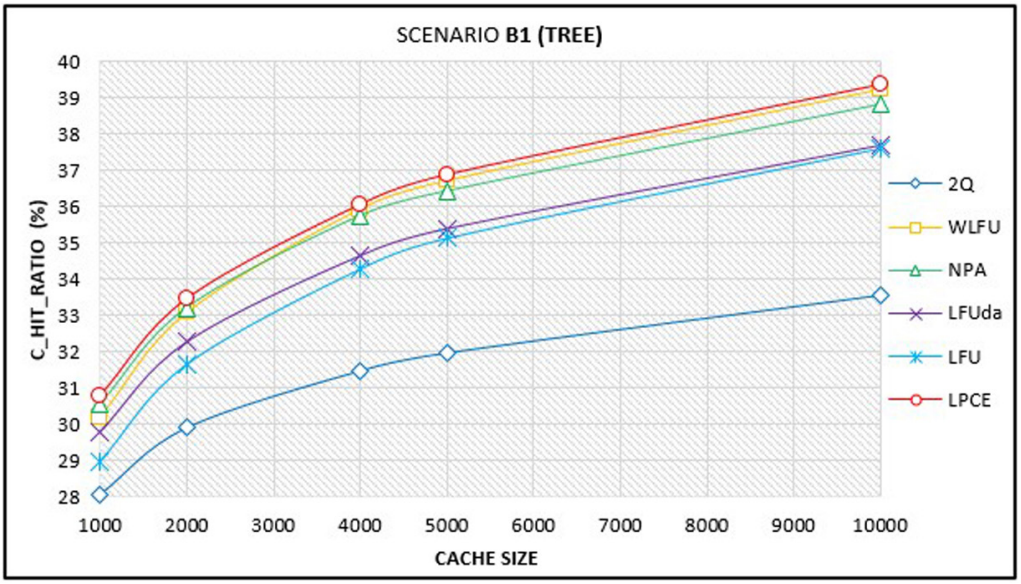

Fig. 12. Cache hit ratio vs. Cache size in scenario B1

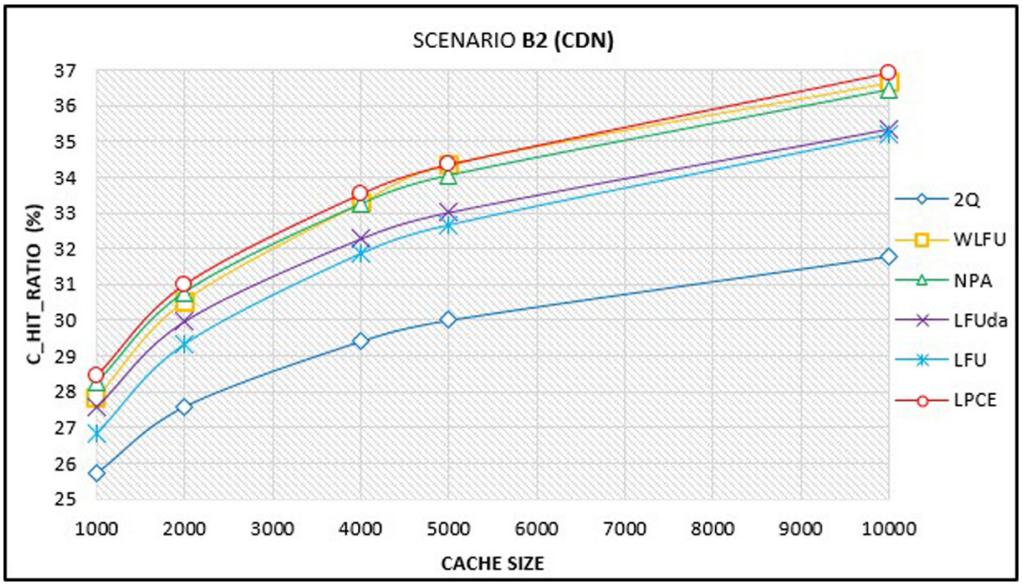

Fig. 13. Cache hit ratio vs. Cache size in scenario B2

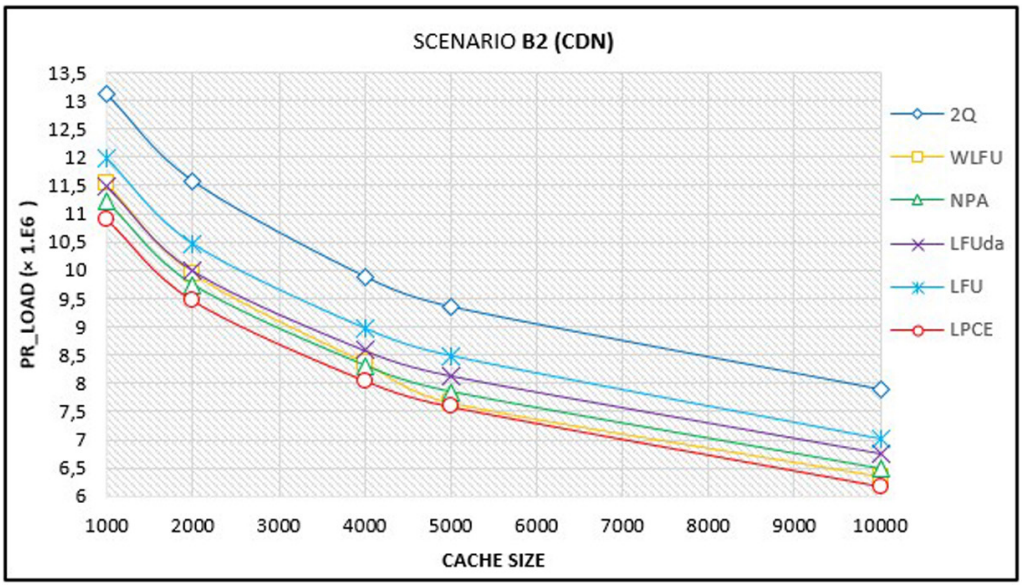

Fig. 14. Producers' load vs. Cache size in scenario B2

In the B2 scenario, Figure 14 shows how the total producers' load changes for LPCE with different cache sizes compared to other policies. LPCE consistently shows the lowest load in all size variations. At a cache size of 1000 contents, LPCE reduces the load on producers by 1.09×10^6 packets compared to LFU, 6.6×10^5 packets compared to WLFU, and 3.3×10^5 packets compared to the NPA policy.

5.3 Scenarios C1 and C2

Figures 15 and 16 show how well LPCE performed compared to other methods in scenarios C1 and C2 as the consumer interest rates changed.

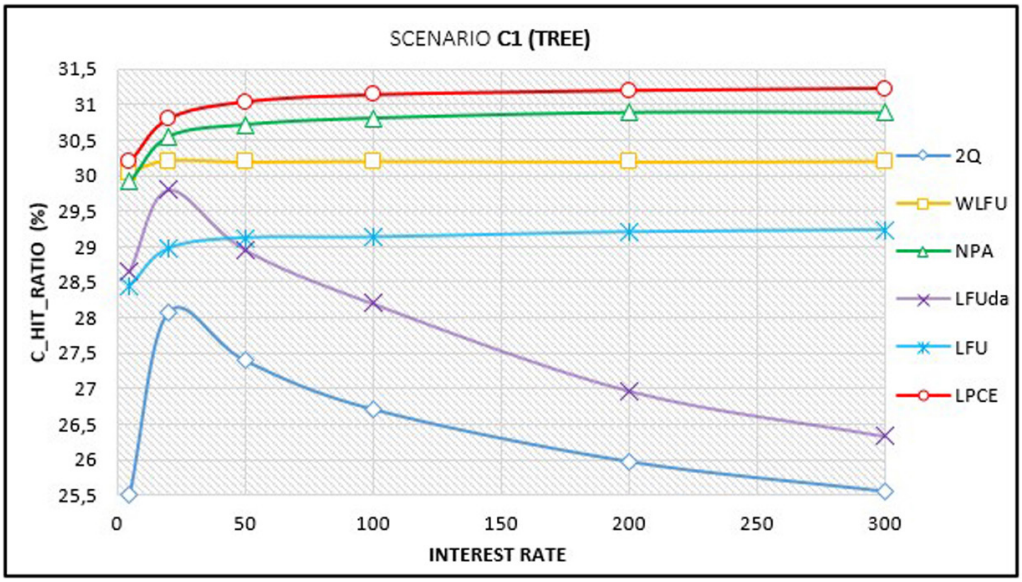

Fig. 15. Cache hit ratio vs. Interest rate in scenario C1

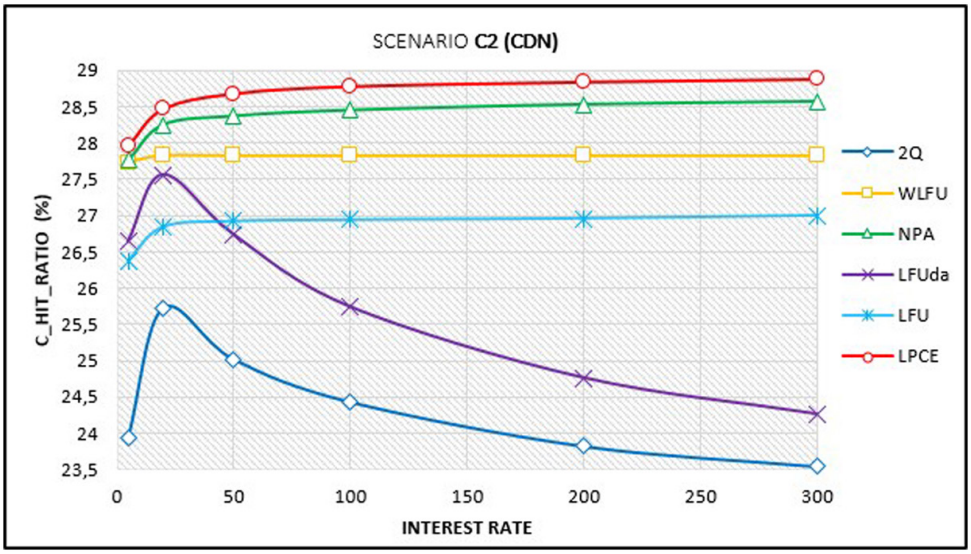

Fig. 16. Cache hit ratio vs. Interest rate in scenario C2

In both scenarios, LPCE responds exceptionally well to the increasing interest rate, exceeding other policies in this aspect. In Figure 15, at a rate of 5 int/s, LPCE excels compared to LFU, displaying a 1.74% improvement in hit rate, surpassing WLFU by 0.15% and the NPA algorithm by 0.27%. At a rate of 300 int/s, LPCE continues its lead over LFU, this time with a 2.0% higher hit rate, once again outperforming WLFU by 1.03% and the NPA algorithm by 0.34%, thereby widening the gap in terms of cache hit rate. An stated differently, as the interest rate increases, the difference in cache hit rate between LPCE and other policies becomes more evident. We can also clearly see how higher interest rates make both the 2Q and LFUda policies less effective.

5.4 Scenarios D1 and D2

In Table 3, we show how the cache hit rate changes for LPCE with different MZipf α values in scenarios D1 and D2. The results clearly demonstrate that LPCE performs significantly better than the other five techniques, regardless of the α value used in both scenarios. As a case in point, in the context of Scenario D1 with α = 1.2, LPCE surpasses LFU, exhibiting a 2.11% increase in hit ratio. It also outperforms 2Q by 2.62%, LFUda by 1.00%, WLFU by 0.66%, and the NPA method by 0.50%. We observe that as the MZipf parameter α value increases, the cache hit ratio improves for all caching policies. This is because a smaller set of contents is requested more frequently with higher α values, resulting in the retrieval of requested contents from the nearest router rather than from the original producer.

Table 3. Cache hit ratio vs. MZipf α in scenarios D1 and D2

	Scenario D1 (Tree)				
α	**0,8**	**0,9**	**1**	**1,1**	**1,2**
2Q	9,45	17,86	28,08	38,34	46,61
WLFU	11,91	20,05	30,21	40,33	48,57
NPA	12,24	20,38	30,55	40,61	48,73
LFUda	11,44	19,65	29,78	40,01	48,23
LFU	10,96	18,98	28,98	39,01	47,12
LPCE	**12,35**	**20,57**	**30,8**	**41,02**	**49,23**

	Scenario D2 (CDN)				
α	**0,8**	**0,9**	**1**	**1,1**	**1,2**
2Q	8,75	16,42	25,72	35,12	42,83
WLFU	11,13	18,56	27,83	37,13	44,76
NPA	11,5	18,95	28,25	37,5	45,06
LFUda	10,73	18,25	27,56	36,98	44,62
LFU	10,32	17,65	26,84	36,08	43,58
LPCE	**11,64**	**19,14**	**28,47**	**37,83**	**45,43**

5.5 Scenarios E1 and E2

In Figures 17 and 18, we can see that LPCE outperformed only four strategies—LFU, 2Q, LFUda, and NPA—as the cache sizes changed. In scenario E1, where the cache size is 10,000 contents (see Figure 17), LPCE excels, surpassing LFU with a 4.35% increase in hit ratio. Furthermore, it exceeds 2Q by 4.47%, LFUda by 0.80%, and the NPA algorithm by 0.73%.

When compared to WLFU, LPCE outperforms it when the cache size is below 5000 contents in both scenarios. For example, in scenario E2 with a cache size of 1000 contents (see Figure 18), LPCE performs better than WLFU with a 0.26% higher hit ratio. Conversely, at a cache size of 10,000 contents, WLFU surpasses LPCE with a 0.31% higher hit ratio. This rule applies exclusively when the interest rate is below 50 int/s. Beyond this threshold, LPCE reclaims the lead and exceeds WLFU comfortably, as will be illustrated later in Figures 22 and 23.

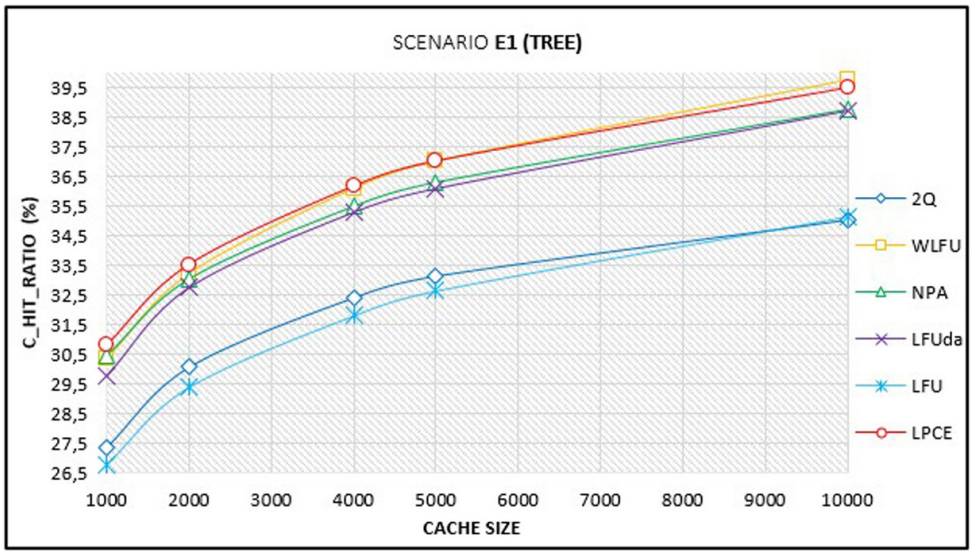

Fig. 17. Cache hit ratio vs. Cache size in scenario E1

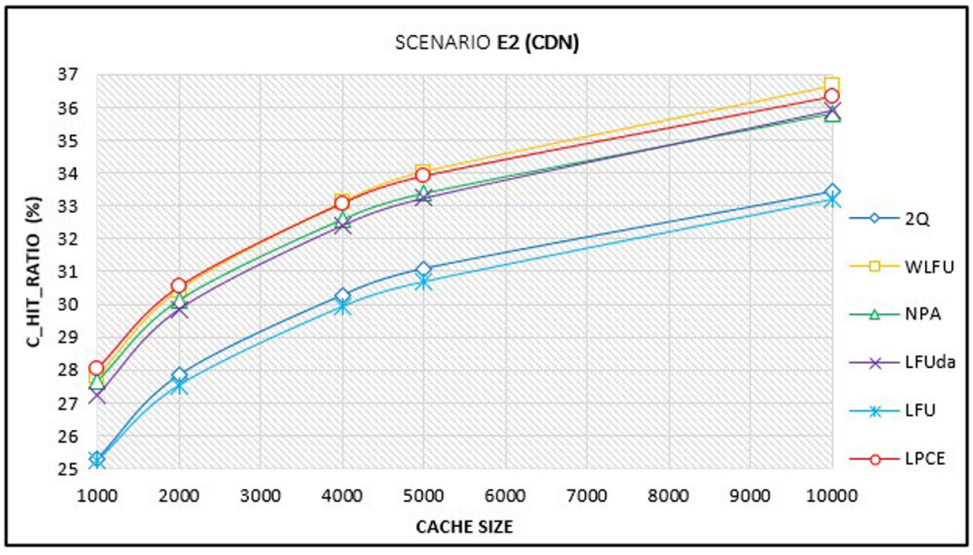

Fig. 18. Cache hit ratio vs. Cache size in scenario E2

Figure 19 illustrates the changes in network traffic with LPCE and different cache sizes in the E2 scenario, compared to other methods. Across different cache sizes, it's evident from the results that LPCE exhibits minimal network traffic compared to only four strategies: LFU, 2Q, LFUda, and NPA. With a cache size of 1000 contents, as an example, LPCE diminishes the network load by 18.10×10^6 packets in comparison to LFU, 4.95×10^6 packets compared to LFUda, and 3.3×10^6 packets compared to the NPA policy. In line with Figures 17 and 18, LPCE shows minimal network traffic when the cache size is less than 5000 contents, as compared to WLFU. As mentioned earlier, this observation holds true only when the interest rate is below 50 int/s.

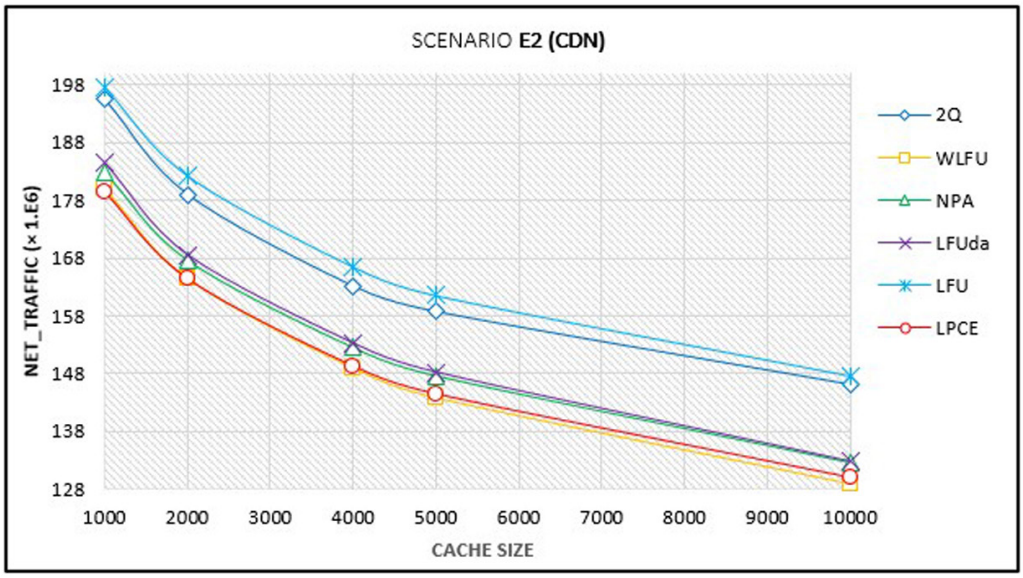

Fig. 19. Network traffic vs. Cache size in scenario E2

5.6 Scenarios F1 and F2

In scenarios F1 and F2, with a cache size of 1000 contents (see Figures 20 and 21), LPCE demonstrates notable responsiveness to the increasing interest rate, surpassing other policies in this regard. In Figure 20, for instance, at an interest rate of 200 int/s, LPCE shows a 3.91% higher hit ratio than LFU and surpasses LFUda by 3.55%.

Remarkably, within the TREE topology, we notice that NPA performs comparably to LPCE when the interest rate exceeds 100 int/s. Nevertheless, when the topology size expands with an increase in both consumers and producers, as in the case of the CDN topology, LPCE regains its advantage and outperforms NPA effortlessly, whatever the interest rate, as shown in Figure 21.

In scenarios E1 and E2, we have mentioned that WLFU outperforms LFU only under the conditions of an interest rate below 50 int/s and a cache size exceeding 5000. Above the 50 int/s threshold, LPCE takes the lead and comfortably surpasses WLFU, regardless of the cache size. Therefore, Figures 22 and 23, depicting scenarios F1 and F2 with a cache size of 10,000 contents, affirm this observation. Despite the cache size exceeding 5000 contents, LPCE manages to surpass WLFU once the interest rate exceeds 50 int/s.

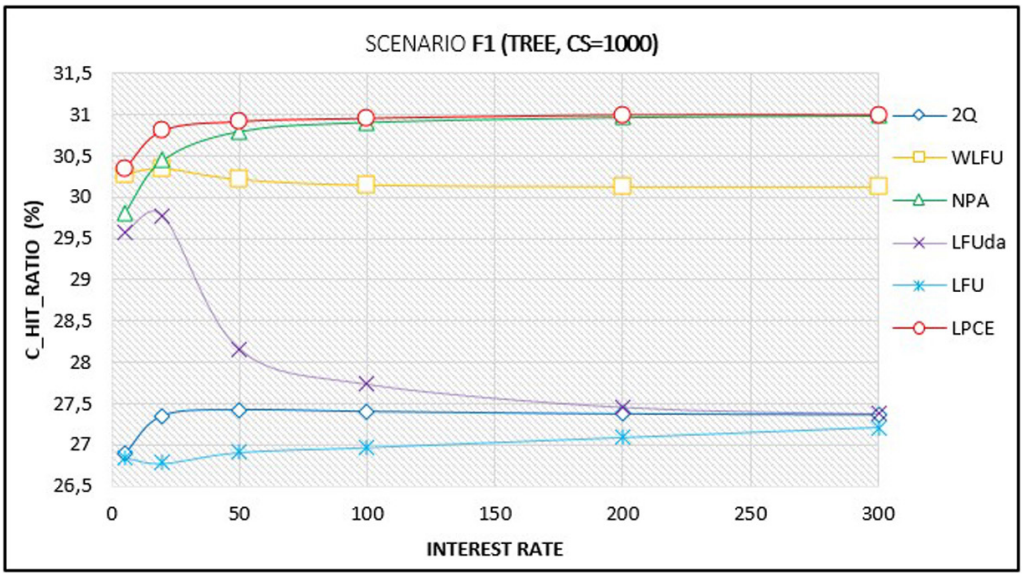

Fig. 20. Cache hit ratio vs. Interest rate in scenario F1

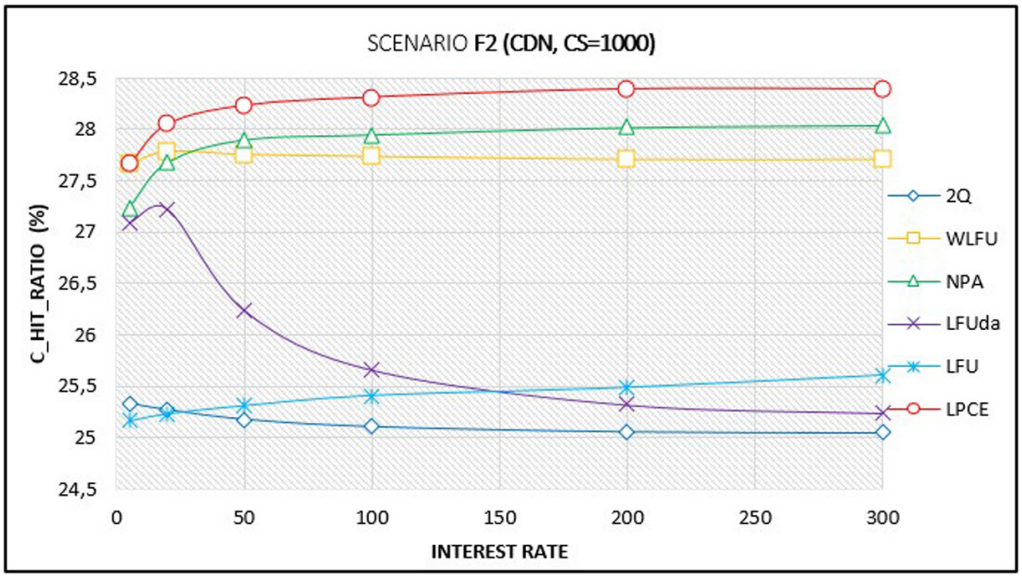

Fig. 21. Cache hit ratio vs. Interest rate in scenario F2

Additionally, in the petite-sized TREE topology containing just one producer and eight consumers, we observe NPA converging towards LPCE as the interest rate reaches 300 int/s (see Figure 22), similar to the scenario depicted in Figure 20, where convergence begins at 100 int/s. However, in the large CDN topology with five producers and thirty-two consumers, LPCE easily recovers its superiority, surpassing NPA in performance (see Figure 23).

Fig. 22. Cache hit ratio vs. Interest rate in scenario F1 with CS = 10000

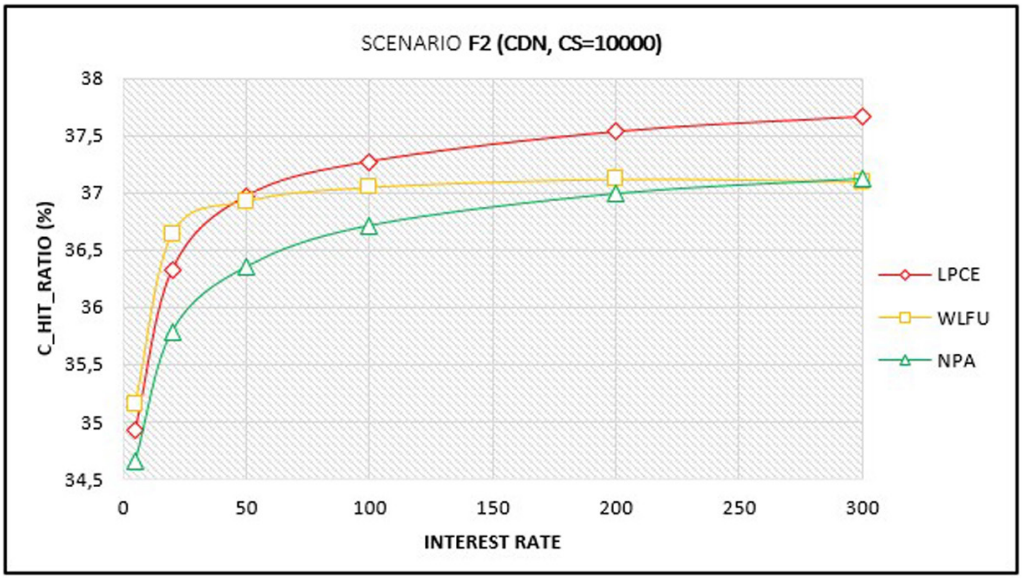

Fig. 23. Cache hit ratio vs. Interest rate in scenario F2 with CS = 10000

5.7 Scenarios G1 and G2

Table 4 illustrates the variations in cache hit rates for LPCE and five other techniques across different MZipf α values in scenarios G1 and G2. The findings consistently show that LPCE achieves the highest cache hit rate, regardless of the α value used in both scenarios. As a demonstration, in scenario G1 with α = 1.2, LPCE exhibits a 5.75% higher hit ratio than LFU. In addition, it outperforms LFUda by 0.81%, NPA by 0.76%, and the WLFU policy by 0.74%.

Table 4. Cache hit ratio vs. MZipf α in scenarios G1 and G2

	Scenario G1 (Tree)				
α	**0,8**	**0,9**	**1**	**1,1**	**1,2**
2Q	10,23	17,76	27,35	37,37	45,65
WLFU	12,31	20,32	30,35	40,37	48,4
NPA	12,26	20,37	30,45	40,47	48,38
LFUda	11,93	19,88	29,78	40,05	48,33
LFU	9,89	17,37	26,78	36,06	43,39
LPCE	**12,33**	**20,55**	**30,81**	**40,96**	**49,14**

	Scenario G1 (CDN)				
α	**0,8**	**0,9**	**1**	**1,1**	**1,2**
2Q	9,68	16,49	25,27	34,36	42,06
WLFU	**11,35**	18,68	27,79	36,93	44,32
NPA	11,1	18,48	27,68	36,88	44,27
LFUda	10,76	18,09	27,22	36,53	44,15
LFU	9,64	16,55	25,23	34,02	40,92
LPCE	11,31	**18,75**	**28,05**	**37,29**	**44,83**

6 CONCLUSION AND FUTURE WORK

The efficiency of caching systems plays a crucial role in determining the performance of NDN networks and has become a focal point of interest in the networking research community. This attention has led to the development and refinement of numerous caching policies for enhancing overall network performance. Key metrics used to evaluate caching performance include cache hit ratio, producer load, network traffic, and content delivery delay. The cache hit rate is the main metric used for assessing performance in NDN. A higher cache hit rate implies that more requests are satisfied from content stores, resulting in lower latency, reduced network traffic, and less strain on original producers.

The performance of caching in NDN networks depends on two critical factors: the strategy used to place content and the policy governing content replacement. In this paper, we introduce the less popular content eviction (LPCE) policy, a novel cache replacement scheme designed to enhance the caching performance of the LFU replacement policy in NDN routers, thus improving the overall network performance. Our replacement approach for Named Data Networking combines LFU and FIFO replacement policies and employs an additional LRU ghost list to preserve the popularity of several recently evicted contents over an extended period.

Using the ccnSim simulator, we evaluated our technique in comparison to the LFU, WLFU, NPA, LFUDA, and 2Q replacement policies. This assessment was based on several scenarios that involve variations in different simulation parameters such as placement strategy, content store size, consumer interest rate, and the number of consumers and producers. Simulation findings showed that the LPCE policy surpassed both the LFU policy and other alternatives in several network metrics, such as cache hit ratio, latency, upstream hop count, producers' load, and network traffic.

Furthermore, under the LCE placement strategy, LPCE improves LFU performance, leading to an increase in cache hit ratio ranging from 1.32% to 2.11%. Similarly, via the LCD placement strategy, LPCE significantly boosts LFU performance, yielding an augmentation in cache hit ratio ranging from 1.67% to 5.75%. Therefore, the proposed LPCE scheme effectively mitigates the impact of aging and establishes itself as a viable alternative to the traditional LFU method.

As future prospects, we aspire to test our method under a distribution pattern that generates content dynamically, in contrast to the MZipf model that follows a static distribution. In addition, we plan to conduct a comparison with ARC, LRFU, LIRS [32], and other advanced caching strategies [33] using real NDN topologies.

7 REFERENCES

[1] J. Roberts, "The clean-slate approach to future internet design: A survey of research initiatives," *Annals of Telecommunications*, vol. 64, pp. 271–276, 2009. https://doi.org/10.1007/s12243-009-0109-y

[2] L. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014. https://doi.org/10.1145/2656877.2656887

[3] A. Anjum, P. Agbaje, A. Mitra, E. Oseghale, E. Nwafor, and H. Olufowobi, "Towards named data networking technology: Emerging applications, use cases, and challenges for secure data communication," *Future Generation Computer Systems*, vol. 151, pp. 12–31, 2024. https://doi.org/10.1016/j.future.2023.09.031

[4] A. V. Vasilakos, Z. Li, G. Simon, and W. You, "Information centric network: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 52, pp. 1–10, 2015. https://doi.org/10.1016/j.jnca.2015.02.001

[5] M. N. D. Satria, F. H. Ilma, and N. R. Syambas, "Performance comparison of named data networking and IP-based networking in Palapa Ring network," in *2017 3rd Int. Conf. on Wireless and Telematics (ICWT)*, vol. 2017, 2018, pp. 43–48. https://doi.org/10.1109/ICWT.2017.8284136

[6] J. Ran, N. Lv, D. Zhang, Y. Ma, and Z. Xie, "On performance of cache policies in named data networking," in *Proceedings of the 2013 International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013)*, 2013, pp. 668–671. https://doi.org/10.2991/icacsei.2013.160

[7] C. Ghasemi, H. Yousefi, and B. Zhang, "Internet-scale video streaming over NDN," *IEEE Network*, vol. 35, no. 5, pp. 174–180, 2021. https://doi.org/10.1109/MNET.121.1900574

[8] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named data networking: A survey," *Computer Science Review*, vol. 19, pp. 15–55, 2016. https://doi.org/10.1016/j.cosrev.2016.01.001

[9] NAMED DATA NETWORKING, "Named Data Networking: Motivation & Details." Available online at: https://named-data.net/project/archoverview/ [Accessed: Feb. 21, 2024].

[10] L. V. Yovita and N. R. Syambas, "Caching on named data network: A survey and future research," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 4456–4466, 2018. https://doi.org/10.11591/ijece.v8i6.pp4456-4466

[11] C. Fan, S. Shannigrahi, C. Papadopoulos, and C. Partridge, "Discovering in-network caching policies in NDN networks from a measurement perspective," in *ICN 2020 Proceedings of the 7th ACM Conference on Information-Centric Networking*, 2020, pp. 106–116. https://doi.org/10.1145/3405656.3418711

[12] 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. https://doi.org/10.1109/COMST.2016.2565541

[13] Y. Gui and Y. Chen, "A cache placement strategy based on compound popularity in named data networking," *IEEE Access*, vol. 8, pp. 196002–196012, 2020. https://doi.org/10.1109/ACCESS.2020.3034329

[14] E. T. d. Silva, J. M. H. de Macedo, and A. L. D. Costa, "NDN content store and caching policies: Performance evaluation," *Computers Journal*, vol. 11, no. 3, p. 37, 2022. https://doi.org/10.3390/computers11030037

[15] N. Laoutarisa, H. Che, and I. Stavrakakisa, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006. https://doi.org/10.1016/j.peva.2005.05.003

[16] M. A. P. Putra, D. S. Kim, and J. M. Lee, "Adaptive LRFU replacement policy for named data network in industrial IoT," *ICT Express*, vol. 8, no. 2, pp. 258–263, 2022. https://doi.org/10.1016/j.icte.2021.10.004

[17] M. Bilal and S.-G. Kang, "A cache management scheme for efficient content eviction and replication in cache networks," *IEEE Access*, vol. 5, pp. 1692–1701, 2017. https://doi.org/10.1109/ACCESS.2017.2669344

[18] E. Aubry, T. Silverston, and I. Chrisment, "Green growth in NDN: Deployment of content stores," in *Proceedings of the 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Rome, Italy, 2016, pp. 1–6. https://doi.org/10.1109/LANMAN.2016.7548850

[19] M. Arlitt, R. Friedrich, and T. Jin, "Performance evaluation of web proxy cache replacement policies," in *Computer Performance Evaluation (TOOLS 1998),* in Lecture Notes in Computer Science, R. Puigjaner, N. N. Savino and B. Serra, Eds., Springer, Berlin, Heidelberg, vol. 1469, 1998, pp. 193–206. https://doi.org/10.1007/3-540-68061-6_16

[20] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, "Evaluating content management techniques for web proxy caches," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, pp. 3–11, 2000. https://doi.org/10.1145/346000.346003

[21] N. R. Syambas, H. Situmorang, and M. A. P. Putra, "Least recently frequently used replacement policy in named data network," in *2019 IEEE 5th International Conference on Wireless and Telematics (ICWT),* Yogyakarta, Indonesia, 2019, pp. 1–4. https://doi.org/10.1109/ICWT47785.2019.8978218

[22] G. Karakostas and D. N. Serpanos, "Exploitation of different types of localities for Web caches," in *Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications,* Taormina-Giardini Naxos, Italy, 2002, pp. 207–212. https://doi.org/10.1109/ISCC.2002.1021680

[23] A. Silva, I. Araujo, N. Linder, and A. Klautau, "Name popularity algorithm: A cache replacement strategy for NDN networks," *Journal of Communication and Information Systems*, vol. 34, no. 1, pp. 206–214, 2019. https://doi.org/10.14209/jcis.2019.22

[24] T. Johnson and D. Shasha, "2Q: A low overhead high performance buffer management replacement algorithm," in *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, Santiago, Chile, 1990, pp. 439–450.

[25] R. M. Negara, N. P. Wasesa, Z. Muhammad, R. Mayasari, and S. Astuti, "Impact of segmentation and popularity based cache replacement policies on named data networking," *Jurnal Rekayasa Elektrika,* vol. 20, no. 1, pp. 35–42, 2024. https://doi.org/10.17529/jre.v20i1.34309

[26] P. Singh, R. Kumar, S. Kannaujia, and N. Sarma, "Adaptive replacement cache policy in named data networking," in *2021 International Conference on Intelligent Technologies (CONIT)*, Hubli, India, 2021, pp. 1–5. https://doi.org/10.1109/CONIT51480.2021.9498489

[27] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnSim: A highly scalable CCN simulator," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 2309–2314. https://doi.org/10.1109/ICC.2013.6654874

[28] A. Vargaand and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (SimuTools 2008)*, Marseille, France, 2008. https://doi.org/10.4108/ICST.SIMUTOOLS2008.3027

[29] C. B. Speranta *et al.*, "Some properties of Zipf's law and applications," *Axioms*, vol. 13, no. 3, p. 146, 2024. https://doi.org/10.3390/axioms13030146

[30] N. E. H. Fethellah, H. Bouziane, and A. Chouarfia, "NECS-based cache management in the information centric networking," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, no. 21, pp. 172–187, 2021. https://doi.org/10.3991/ijim.v15i21.20011

[31] S. Al-Ahmadi, "A new efficient cache replacement strategy for named data networking," *International Journal of Computer Networks and Communications*, vol. 13, no. 5, pp. 19–35, 2021. https://doi.org/10.5121/ijcnc.2021.13502

[32] X. Weiand and Y. Liu, "Low inter-reference recency set replacement policy in named data networking," in *2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*, Chongqing, China, 2020, pp. 1155–1160. https://doi.org/10.1109/ICIBA50161.2020.9277372

[33] R. Alubady, M. Salman, and A. S. Mohamed, "A review of modern caching strategies in named data network: Overview, classification, and research directions," *Telecommun. Syst.*, vol. 84, pp. 581–626, 2023. https://doi.org/10.1007/s11235-023-01015-3

8 AUTHORS

Samir Nassane received his MSc in Networks and Databases from the University of Science and Technology of Oran, Algeria, in 2007. He has held the position of Assistant Professor in the Department of Computer Sciences at the University of Tiaret, Algeria, since 2009. Now, he is pursuing a Ph.D. in Intelligent Systems and is a member of the Laboratory of Energetic Engineering and Computer Engineering (L2GEGI) at the same university. His expertise includes multi-agent systems, ad hoc networks, and named data networking (NDN) (E-mail: samir.nassane@univ-tiaret.dz).

Sid Ahmed Mokhtar Mostefaoui is currently an Associate Professor at the University of Tiaret, Algeria. He serves as the Director of the Research Laboratory in Artificial Intelligence and Systems (LRIAS) at the same university. His main research interests include intelligent systems, adaptive control systems in smart spaces, and the new generation of internet networks (NDN). He obtained his Engineering diploma from the University of Science and Technology of Oran, Algeria, in 2002. Then, he received his MSc in Computer Information Systems from the University of Tiaret in 2010. Later, in 2016, he was awarded a Doctorate degree from the University of Science and Technology of Oran, Algeria (E-mail: mokhtar.mostefaoui@univ-tiaret.dz).

Bendaoud Mebarek received a Ph.D. degree from the Polytechnic National School of Oran and is currently affiliated with the Department of Computer Sciences at the University of Tiaret, he is also a member of the Research Laboratory in Artificial Intelligence and Systems (LRIAS) at the same university. His areas of expertise include modeling and simulation, computational physics, and artificial intelligence. He has authored and presented numerous articles and conferences in reputable journals (E-mail: bendaoud.mebarek@univ-tiaret.dz).

Abdelkader Alem received his Engineering degree in Computer Science from the University of Tiaret, Algeria, in 2006. Currently, he is a Ph.D. candidate in Intelligent Systems and is a member of the Laboratory of Energetic Engineering and Computer Engineering (L2GEGI) at the same university. His research interests include network security, Internet of Things (IoTs), and NDN networks (E-mail: abdelkader83.alem@univ-tiaret.dz).