applied sciences

MDPI

Article

IMU: A Content Replacement Policy for CCN, Based on Immature Content Selection

Salman Rashid *, Shukor Abd Razak * and Fuad A. Ghaleb *

School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia
* Correspondence: rashid.salman@graduate.utm.my (S.R.); shukorar@utm.my (S.A.R.); abdulgaleel@utm.my (F.A.G.)

Abstract: In-network caching is the essential part of Content-Centric Networking (CCN). The main aim of a CCN caching module is data distribution within the network. Each CCN node can cache content according to its placement policy. Therefore, it is fully equipped to meet the requirements of future networks demands. The placement strategy decides to cache the content at the optimized location and minimize content redundancy within the network. When cache capacity is full, the content eviction policy decides which content should stay in the cache and which content should be evicted. Hence, network performance and cache hit ratio almost equally depend on the content placement and replacement policies. Content eviction policies have diverse requirements due to limited cache capacity, higher request rates, and the rapid change of cache states. Many replacement policies follow the concept of low or high popularity and data freshness for content eviction. However, when content loses its popularity after becoming very popular in a certain period, it remains in the cache space. Moreover, content is evicted from the cache space before it becomes popular. To handle the above-mentioned issue, we introduced the concept of maturity/immaturity of the content. The proposed policy, named Immature Used (IMU), finds the content maturity index by using the content arrival time and its frequency within a specific time frame. Also, it determines the maturity level through a maturity classifier. In the case of a full cache, the least immature content is evicted from the cache space. We performed extensive simulations in the simulator (Icarus) to evaluate the performance (cache hit ratio, path stretch, latency, and link load) of the proposed policy with different well-known cache replacement policies in CCN. The obtained results, with varying popularity and cache sizes, indicate that our proposed policy can achieve up to 14.31% more cache hits, 5.91% reduced latency, 3.82% improved path stretch, and 9.53% decreased link load, compared to the recently proposed technique. Moreover, the proposed policy performed significantly better compared to other baseline approaches.

Keywords: content replacement; content placement; content-centric networking; cache networks; immaturity; stretch reduction

check for updates

Citation: Rashid, S.; Razak, S.A.; Ghaleb, F.A. IMU: A Content Replacement Policy for CCN, Based on Immature Content Selection. *Appl. Sci.* **2022**, *12*, 344. https://doi.org/10.3390/app12010344

Academic Editor: Yangquan Chen

Received: 18 November 2021
Accepted: 27 December 2021
Published: 30 December 2021

1. Introduction

Due to advancements in technology, things are becoming more integrated and intelligent, leading to a rapid increase in Internet usability. Internet usage patterns demonstrate that new era applications are becoming more sensitive in bandwidth and latency. IP video traffic is expected to dominate overall IP traffic by 82% by 2022 [1], up from 74% in 2017 [2]. Internet users are not interested in the location of the storage server. Their primary interest is having Internet connectivity that assures fast and reliable retrieval of desired information. Content-centric networking (CCN) has proven to be a promising solution to meet the needs of future networks [3]. CCN naturally supports in-network caching and it attempts to respond to the requested data when a user request contains the name or identity of the desired data. CCN assigns each piece of data a unique identity and addresses data objects at the network level, in contrast to the Internet's host-centric architecture. CCN naturally supports in-network caching and many-to-many communication [4]. When the user request contains the name or identification of the desired data object, the network attempts to respond to the request with the data object. The name can also belong to a location or a host machine. This mechanism makes CCN more general than the host-to-host communication model [5].

In-network caching provides a solution for traditional Internet architecture that works in the application layer [3]. The content of the CCN cache changes rapidly due to enormous data demands. Furthermore, CCN is a solution that works on the network layer level [6]. It allows the CCN node to cache a temporary copy of the requested content. CCN can minimize network traffic and maximize content availability by providing the desired content closer to the consumer [7]; it is difficult to decide the cache location of the content to satisfy consumer requests and improve network performance [8]. In addition, it is also important to determine which content should be removed from the cache space to accommodate new content in the cache. Improper content selection causes the degradation of network performance [9]. In-network caching faces several challenges, including limited cache storage, caching replacement and placement, caching traffic, and complex network topology [10,11].

The performance of the CCN depends on the content placement and replacement policies. The content placement policy decides the appropriate cache location of each content [1]. Hence, the node selection for content caching should be optimized to satisfy consumer requests, with minimum overhead. Due to the limited cache capacity in the node, any cached content in the cache needs to be removed to accommodate new content [12]. Content replacement policy is responsible for choosing the right content against defined criteria [13]. The network performance and cache hit ratio decreases if popular content is removed from the cache or unpopular content remains in cache for a long time [14–17].

Although cached contents at all nodes, along with the routing path, increase the network performance and cache hit ratio, it is not a practical approach due to the finite cache space. That is, if the cache space is full and new content arrives, one of the cached content needs to be removed from the cache space. However, most existing replacement policies follow the concept of the Least Frequently Used (LFU) or Least Recently Used (LRU) policy to replace the content which is not effective for CNN [18]. The newly arrived content may become popular over time due to high demand. When popular content loses its popularity, it stays in the cache due to previous popularity. Therefore, the network performance may decrease due to the overstay of previous popular content that is currently unpopular or the eviction of currently popular content. To solve this issue and improve network performance, we introduced a new concept of content maturity and immaturity to deal with the aforementioned issues. The content that loses its popularity over a specific time frame and stays in the cache for a long time is called immature content. In contrast, the content will be considered mature if it has high popularity and is also recently requested in the network within a specific time frame. Every new content is neither popular nor mature. Content should stay in the cache for some time to know its maturity level. Hence, such content is not evicted from the cache, which is yet to become popular. In addition, this concept removes content from the cache that loses its popularity after being highly popular for some time.

A content replacement policy is proposed in this work called IMU (Immaturity used). This policy removes content from the cache that is immature within a limited time frame. Therefore, most of the contents in the cache are recently used and highly popular, leading to a better cache hit ratio and network performance. The key contributions are summarized below:

- A new concept of content maturity/immaturity has been introduced to design and develop an effective content eviction policy. The proposed content eviction policy evicts the content from cache through the immature content selection to improve the cache hit ratio, latency, path stretch, and link load.

- A mechanism to calculate the maturity level of the content has been designed and developed using the content frequency and arrival time of the content.
- Icarus [19] has been used to verify the performance of the proposed policy with existing state-of-the-art content replacement policies. Subsequently, we gained substantial improvement in cache hit ratio, latency, path stretch, and link load.

The rest of this paper is organized as follows: We discuss the related work in Section 2. The proposed policy is described in Section 3, which highlights its contribution. Section 4 describes the simulation environment and parameters as well as the result analysis and discussion. Finally, the conclusion and future work are in Section 5.

2. Related Work

Content eviction policy works when the cache space is filled with content. The eviction policy provides a mechanism to replace existing contents with requested contents in the cache. The eviction policy must keep popular contents in the cache with the least processing complexity. In general, an eviction policy should have two properties. First, the eviction policy should not remove popular content from the cache. Second, it also keeps the most frequently used contents in the cache by applying some sort of priority. Several eviction policies were proposed in the past [9,19–25]. Some of the most popular eviction policies include First in First out (FIFO), Random Replacement (RR), Least Recently Used (LRU), Least Frequently Used (LFU), Window-LFU (W-LFU), Least Frequent Recently Used (LFRU), Popularity Prediction Caching (PPC), Network-oriented Information-centric Centrality for Efficiency (NICE), NC-SDN, and Least Fresh First (LFF). A brief description of each cache eviction policy is summarized below.

As the name suggests, FIFO replaces the content from the cache based on a first-come, first-serve basis. The content item that comes first in the cache is evicted first when there is a need for replacement [20]. It does not deal with the importance or priority of the content being replaced by the new content. RR policy randomly selects existing content from the cache to replace it with new content [21]. However, it has no particular criteria for content selection from the cache. LRU is a typical policy that has extensive usage in cache eviction [22]. LRU keeps track of the usage of each content in cache. When the replacement request is received, LRU checks the requested content in the cache. If this requested content is not already in the cache, it evicts the least recently used content to accommodate the requested content. Therefore, LRU is simple to implement and has less computational delay.. But on the other side, LRU does not consider the content frequency (dynamic changes of popularity over time), which plays a significant role in network performance and the cache hit ratio.

LFU keeps track of the frequency of each content in the cache [23]. LFU serves to store the most popular content in the cache statically. It keeps a counter of how many times the content is requested. Whenever a request is received for content, the counter value is incremented by one. When the cache space is full and there is a requirement to replace content, the content with the least counter value is selected to evict. LFU keeps popular content in the cache, but it requires a very high processing time that leads to performance degradation in CCN. Further, when content that has been popular for some time loses its popularity, it stays in the cache, causing severe performance losses. W-LFU is an eviction policy that uses a limited number of access requests over a time window [24]. This technique tries to solve the LFU problem by keeping the history of the requested contents. This record of history is referred to as a window. The size of this window is directly proportional to the total number of contents and the cache size in the network. This policy demonstrates considerable improvements, but it fails to evict suitable content in the case of bursty requests. Moreover, this policy only observes a small portion of the cache, making it impractical for full cache capacity.

LFRU is the combination of LRU and LFU [25]. According to the LFRU eviction policy, a cache divides into unprivileged and privileged partitions. The privileged partition is known as a protected partition. The popular content is pushed into the privileged partition.

If the privileged partition is fully occupied and there is no more space available to store content, the LFRU ensures that the content is evicted from the unprivileged partition and that content is transferred from the privileged to the unprivileged partition. Filtering out the locally popular contents and placing the popular contents in the privileged partition are the key features of the LFRU eviction policy. This policy demonstrates considerable improvements, but it fails to evict suitable content in the case of bursty requests. Moreover, this technique also requires a large processing time to manage partitions.

PPC is a chunk-level in-network caching eviction policy [26]. It is capable of predicting the popularity of the video chunks. PPC stores content based on the popularity that it predicts. On the other side, the contents that have the least popular prediction are evicted out. This eviction scheme is also termed the Assist-predict model. It is based on the request information of the neighboring chunks. It also predicts future popularity by using past experience with the popularity of the content. If the popularity of the new content is less than the former popularity, the newly incoming chunk does not cache. Otherwise, it evicts the future content based on popular prediction. This model-based prediction technique works well but fails to predict properly against frequently changing requests. Moreover, this policy leads to high network load due to control signaling overhead and high computational workload. The NC-SDN eviction model was introduced as a cache eviction algorithm that relies on SDN (software-defined networking) [16]. NC-SDN model uses three arguments. First, it calculates data popularity; second, it comes to know the location of cache management switches; third, it facilitates cooperation among different nodes in the network. When the cache is fully occupied, it checks the popularity of each content and replaces the least popular content with new content. Although the replacement technique is straightforward, the control traffic and exchange of information between the switches are very high, leading to performance losses.

LFF is a content replacement policy that predicts the time of the next event [27]. Based on the prediction, it controls the residual life of retrieved content. When the cache capacity is full, this policy measures the time for which the content is considered invalid. In addition, this policy checks whether the source has been updated after retrieving the content to check the validity of each content. This policy ignores the high replacement rate in the central node and does excessive computing, making it impractical for large CCN. NICE has been introduced as a new metric for cache management in ICN [28]. This policy uses a method that computes the centrality. Centralization is used in the replacement phase to manage cache contents. This method is based on the number of caches instead of the number of contents. Content is replaced when the NICE value is high, as the contents move from one cache to the other due to the centrality of the content. However, it causes high network load and computational complexity.

Most of the replacement strategies [27–34] on CCN focus on content frequency, popularity, and time freshness. These policies ignore the concept of content immaturity in content eviction; it is neither popular nor mature when new content is cached in the cache. We need some time to evaluate whether this content has become popular or not. If that content is removed from the cache, the consumer has to retrieve it from the publisher, which affects network performance. Therefore, content may become popular for a certain period, and then its popularity starts decreasing [29]; if that content is not removed from the cache, network performance and the cache hit ratio also degrade. When the cache space is low and the popularity of the content changes frequently, it becomes challenging for the content eviction policy to decide which content should be evicted from the cache space. A content eviction policy should be able to provide equal opportunities for each content to become mature. Therefore, we introduced a concept of maturity and immaturity of the content, and our proposed cache replacement policy uses this concept to accommodate the request of new content. The proposed policy evicts immature content to solve content popularity issues.

3. Proposed Content Replacement Policy

Content replacement policy is an integral part of CCN cache management. The nodes in CCN need to free up space over time, due to limited cache space, so that new contents are cached. It is a crucial decision to evict content from the cache, which, in turn, increases or decreases network performance and the cache hit ratio. Numerous content replacement policies decide to evict content from the cache using various criteria, such as time in the cache, frequency, popularity, and node centralization. These policies do not use content immaturity for eviction. The proposed policy selects immature content from the cache that stays for a long time in the cache and has a lower frequency in a particular time window. Thus, the proposed policy avoids unnecessary content occupation in the cache space. Due to immature content eviction, network nodes contain more requested content within the cache space. Therefore, more customer interests are satisfied within the network.

The proposed technique determines the mature/immature contents. Algorithm 1 elaborates the procedure to label a content, s_i is mature or immature. The proposed policy keeps track of each content's arrival time and frequency at each node. The current time and the frequency of the node s_i is denoted by $\text{Ꚋ}_{c\acute{r},s_i}$ and $\text{Ƒ}_{\check{c},s_i}$ respectively. The proposed technique calculates the content period, $\text{Ʈ}_{p,s_i}$, with the help of content frequency, $\text{Ƒ}_{\check{c},s_i}$, and content arrival time, $\text{Ꚋ}_{c\acute{r},s_i}$. Therefore, it determines the duration of the content s_i in the cache space. Then, the proposed policy calculates the maturity index $\text{Ϻ}_{c\ddot{I},s_i}$ by dividing the frequency of the content $\text{Ƒ}_{\check{c},s_i}$ and content period $\text{Ʈ}_{p,s_i}$. The maturity classifier $\text{Ϻ}_{Ł}$ is calculated using the median of maturity indexes $\text{Ϻ}_{Ł} \leftarrow \mathit{Median}\left(\text{Ϻ}_{c\ddot{I},s_n}\right)$. Content s_i whose maturity index $\text{Ϻ}_{c\ddot{I},s_i}$ exceeds the value of $\text{Ϻ}_{Ł}$ is classified as mature content; otherwise, it is immature content. The median is used for finding the relevant mean value of the maturity index $\text{Ϻ}_{c\ddot{I},s_i}$, because it is not affected by lower or extreme high set values. Thus, this provides a fair value to the maturity classifier $\text{Ϻ}_{Ł}$.

Algorithm 1: Determine the mature and immature content.

```
Input: Suppose S ∈ {s_1, s_2, s_3, ..., s_n} is set of contents.
Output: Categorization of contents.
$\text{Ꚋ}_{c\acute{r},s_i}$   is the arrival time of ith content.
$\text{Ƒ}_{\check{c},s_i}$   is the frequency of ith content.
$W_T$   is the size of the time window.
$\text{Ʈ}_{p,s_i}$   is the time period of ith content.
$\text{Ϻ}_{c\ddot{I},s_i}$   is the maturity index of ith content.
$\text{Ϻ}_{Ł}$   is the maturity classifier.
1. for $i = 1 : n$
        $\text{Ʈ}_{p,s_i} \leftarrow W_T - \text{Ꚋ}_{c\acute{r},s_i}$
        $\text{Ϻ}_{c\ddot{I},s_i} \leftarrow \text{Ƒ}_{\check{c},s_i} / \text{Ʈ}_{p,s_i}$
2. $\text{Ϻ}_{Ł} \leftarrow \mathit{Median}\left(\text{Ϻ}_{c\ddot{I},s_n}\right)$
3. for $i = 1 : n$
     if    $\text{Ϻ}_{c\ddot{I},s_i} >= \text{Ϻ}_{Ł}$
              $s_i$ is mature
     else
              $s_i$ is immature
```

Algorithm 2 describes the next part of our proposed policy. When a node v receives an interest packet for content s_i, and the time window has not expired, then the proposed policy finds the requested content s_i in the local cache. In the case of a cache hit, the proposed policy increments the frequency of content s_i by one and associates a new arrival time $\text{Ꚋ}_{c\acute{r},s_i}$. Moreover, node v discards the interest packet from PIT and replies through the data packet to the requested consumer. Otherwise, a cache miss means that the requested content s_i is being cached for the first time in CS. Thus, its frequency $Ƒ_{č,s_i}$ is one and it is associated with the current timestamp $Ƭ_{cŕ,s_i}$. When the cache is full, it selects content s_k with a minimum value of the maturity index $\mathrm{M}_{cĭ,s_i}$ and evicts it from the cache space. Then, the proposed technique checks the time window W_T ; if W_T is expired, then the frequency of all content $Ƒ_{č,s_n}$ is set to one, and the previously associated timestamp $Ƭ_{cŕ,s_n}$ remains the same.

Algorithm 2: IMU Replacement Policy.

```
Input: Request for a content $s_i$ at node $v$
Output: Content selection for replacement of newly arrived content
1. if $W_T$ is not expired
        check local cache
            if cache hit
                $Ƒ_{č,s_i} \leftarrow Ƒ_{č,s_i} + 1$
                $Ƭ_{cŕ,s_i} \leftarrow$ current time
            else if cahe_size == full
                $s_k \leftarrow$ select the content with min $\mathrm{M}_{cĭ,s_i}$
                evict $s_k$
                place $s_i$ in cache
                $Ƒ_{č,s_i} \leftarrow 1$
                $Ƭ_{cŕ,s_i} \leftarrow$ current time
            else
                place $s_i$ in cache
                $Ƒ_{č,s_i} \leftarrow 1$
                $Ƭ_{cŕ,s_i} \leftarrow$ current time
2. else
            for each $s_i$
                $Ƒ_{č,s_i} \leftarrow 1$
            Update $W_T$
            go to step 1
```

For simplicity, we assume that all the CCN based routers (node) have the same cache sizes, cached content, and discrete instants of time for interests to arrive. CS is the local cache size, and the window size is denoted by W_T . There are some events related to content s_i, including received interest packet, received data packet, reply data packet, forward interest packet, cached content, eviction from the cache, and look-up content in local CS. The received interest packet (RIP), received data packet (RDP), reply data packet (REDP), forward interest packet (FIP), cached content (CC), eviction from the cache (EC), and look-up content (LU) are denoted by $Ř_{s_{ip}}$, $Ŕ_{s_{dp}}$, $R̀_{s_{edp}}$, $Ḟ_{s_{ip}}$, $Ḉ_s$, $È_s$ and $Ĺ_s$, respectively. These notations are helpful to understand the whole process of the proposed policy. For example, initially, we assumed the value of cache space (CS) = 6, W_T = 4 s, t = [1, 2, 3, 4, . . . , 13], S ∈ {A, B, C, D, E, F, G, H, I}, as presented in Figure 1.

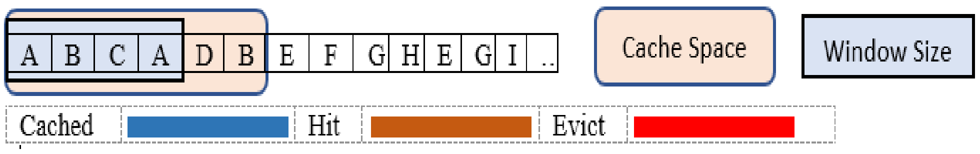


Figure 1. Node cache space, window size, and caching events.

The consumer's requested content (RC) is in the same sequence, window size, and cache space illustrated in Figure 1. The colors indicate three caching processes: cached, hit, and evicted content from the cache. With the help of these colors, we can easily understand the new entry in our tables and variations in the values.

We assume that the cache of a node is empty. The detailed caching process at t = 1 to t = 4 is expressed in Table 1, and Table 2 also maps the IMU process with values. Moreover, we see the effect of values after the window W_T is not expired. The router receives an interest packet for content A $Ř_{A_{ip}}$ but does not find that content in its CS after look-up $Ĺ_A$, then routers the same interest packet forwards $Ḟ_{A_{ip}}$ to the next router. The next router has A content and responds through the data packet $Ȑ_{A_{edp}}$, from which it receives the packet of interest. Furthermore, the router received A data packet $Ŕ_{A_{dp}}$ is then cached $Ḉ_A$ in CS, along with values $Ƭ_{cŕ,s_A}$ = 1, $Ƒ_{č,s_A}$ = 1, $Ț_{p,s_A}$ = 4, and $M_{cÏ,s_A}$ = 0.25. This process will be the same for content B and C. When a hit occurs at t = 4, then the values of $Ƭ_{cŕ,s_A}$ = 4, $Ƒ_{č,s_A}$ = 2, $Ț_{p,s_A}$ = 1, and $M_{cÏ,s_A}$ = 2. Furthermore, window W_T has expired at the same t = 4, but the cache space is not yet full. Then, the router receives an interest packet for content A $Ř_{A_{ip}}$. The hit occurred at t = 4 and changed the values of values $Ƭ_{cŕ,s_A}$ = 4, $Ƒ_{č,s_A}$ = 2, $Ț_{p,s_A}$ = 1, and $M_{cÏ,s_A}$ = 2.00.

Table 1. Caching process at t = 1 to t = 4.

Time t	RC s_i	RIP $Ř_{s_{ip}}$	LU $Ĺ_s$	FIP $Ḟ_{s_{ip}}$	REDP $Ȑ_{s_{edp}}$	RDP $Ŕ_{s_{dp}}$	CC $Ḉ_s$	EC $È_s$
1	A	$Ř_{A_{ip}}$ (1)	$Ĺ_A$ (2)	$Ḟ_{A_{ip}}$ (3)	$Ȑ_{A_{edp}}$ (4)	$Ŕ_{A_{dp}}$ (5)	$Ḉ_A$ (6)	
2	B	$Ř_{B_{ip}}$ (1)	$Ĺ_B$ (2)	$Ḟ_{B_{ip}}$ (3)	$Ȑ_{B_{edp}}$ (4)	$Ŕ_{B_{dp}}$ (5)	$Ḉ_B$ (6)	
3	C	$Ř_{C_{ip}}$ (1)	$Ĺ_C$ (2)	$Ḟ_{C_{ip}}$ (3)	$Ȑ_{C_{edp}}$ (4)	$Ŕ_{B_{dp}}$ (5)	$Ḉ_C$ (6)	
4	A	$Ř_{A_{ip}}$ (1)	$Ĺ_A$ (2)		$Ȑ_{A_{edp}}$ (3)			

Table 2. IMU process at t = 1 to t = 4.

Time	s_i	$Ƭ_{cŕ,s_i}$	$Ƒ_{č,s_i}$	$Ț_{p,s_i}$	$M_{cÏ,s_i}$
t = 1	A	1	1	4	0.25
t = 2	A	1	1	4	0.25
	B	2	1	3	0.33
t = 3	A	1	1	4	0.25
	B	2	1	3	0.33
	C	3	1	2	0.50
t = 4	A	4	2	1	2.00
	B	2	1	3	0.33
	C	3	1	2	0.50

Table 3 describes the detailed caching process at t = 5 to t = 8, and Table 4 also maps the IMU process with values. Table 4 demonstrates that the new window W_T starts at t = 5, and that all the values of $Ƒ_{č,s_i}$ become 1 and retain all the values of $Ƭ_{cŕ,s_i}$. Content D has cached $Ḉ_D$ at t = 5 and displays the values of $Ƒ_{č,s_D}$ = 1, $Ƭ_{cŕ,s_D}$ = 5, $Ț_{p,s_D}$ = 4, and $M_{cÏ,s_D}$ = 0.25 in Table 4. Furthermore, at this stage, cache space CS = 4. After the hit occurs at t = 6, the values of $T_{c\acute{r},s_D} = 6$, $F_{\check{c},s_D} = 2$, $T_{p,s_D} = 3$, and $M_{c\ddot{I},s_D} = 0.67$ have changed. New contents are cached at t = 7 and t = 8, $Ḉ_E$ and $Ḉ_F$, respectively.

Table 3. Caching process at t = 5 to t = 8.

Time **t**	**RC** s_i	**RIP** $\check{R}_{s_{ip}}$	**LU** $\acute{L}_s$	**FIP** $\dot{F}_{s_{ip}}$	**REDP** $\ddot{R}_{s_{edp}}$	**RDP** $\acute{R}_{s_{dp}}$	**CC** $Ḉ_s$	**EC** $\grave{E}_s$
5	D	$\check{R}_{D_{ip}}$ (**1**)	$\acute{L}_D$ (**2**)	$\dot{F}_{D_{ip}}$ (**3**)	$\ddot{R}_{D_{edp}}$ (**4**)	$\acute{R}_{D_{dp}}$ (**5**)	$Ḉ_D$ (**6**)	
6	B	$\check{R}_{B_{ip}}$ (**1**)	$\acute{L}_B$ (**2**)		$\ddot{R}_{B_{edp}}$ (**3**)			
7	E	$\check{R}_{E_{ip}}$ (**1**)	$\acute{L}_E$ (**2**)	$\dot{F}_{E_{ip}}$ (**3**)	$\ddot{R}_{E_{edp}}$ (**4**)	$\acute{R}_{E_{dp}}$ (**5**)	$Ḉ_E$ (**6**)	
8	F	$\check{R}_{F_{ip}}$ (**1**)	$\acute{L}_F$ (**2**)	$\dot{F}_{F_{ip}}$ (**3**)	$\ddot{R}_{F_{edp}}$ (**4**)	$\acute{R}_{F_{dp}}$ (**5**)	$Ḉ_F$ (**6**)	

Table 4. IMU process at t = 5 to t = 8.

Time	s_i	$T_{c\acute{r},s_i}$	$F_{\check{c},s_i}$	T_{p,s_i}	$M_{c\ddot{I},s_i}$	**Time**	s_i	$T_{c\acute{r},s_i}$	$F_{\check{c},s_i}$	T_{p,s_i}	$M_{c\ddot{I},s_i}$
t = 5	A	4	1	5	0.20	t = 7	B	6	2	3	0.67
	B	2	1	7	0.14		A	4	1	5	0.20
	C	3	1	6	0.17		C	3	1	6	0.17
	D	5	1	4	0.25		D	5	1	4	0.25
t = 6	B	6	2	3	0.67		E	7	1	2	0.50
	A	4	1	5	0.20	t = 8	B	6	2	3	0.67
	C	3	1	6	0.17		A	4	1	5	0.20
	D	5	1	4	0.25		C	3	1	6	0.17
							D	5	1	4	0.25
							E	7	1	2	0.50
							F	8	1	1	1.00

Content E and F have cached $Ḉ_E$ and $Ḉ_F$, respectively, at t = 6 and t = 7, and the new values are presented in Table 4. We see that the caching process is displayed step by step in Table 3, and the numbers are associated with each process to illustrate the sequence of this process.

Table 5 reflects caching events from t = 9 to t = 12. At t = 9, the router receives an interest packet of G $\check{R}_{G_{ip}}$. After the look-up $\acute{L}_G$ content is not found in CS, the interest packet is forwarded $\dot{F}_{G_{ip}}$ to the next router. This time, CS is full when it receives the $\acute{R}_{G_{dp}}$ data packet. Now, we find the lowest $M_{c\ddot{I},s_C} = 0.10$ value and remove that content $\grave{E}_C$ from CS. Therefore, it caches the new content $Ḉ_G$ with the associated values of $T_{c\acute{r},s_G} = 9$, $F_{\check{c},s_G} = 1$, $T_{p,s_G} = 4$, and $M_{c\ddot{I},s_G} = 0.25$ in the CS. We can observe in Table 6 how the IMU works when the memory is full and new content arrives simultaneously.

We repeat the same process at t = 10 for content H. The hit occurs at t = 11, and t = 12 updates the values of $T_{c\acute{r},s_E}$, $F_{\check{c},s_E}$, T_{p,s_E}, and $M_{c\ddot{I},s_E}$, as illustrated in Table 6. The hit occurred at t = 11 and t = 12 for requested contents E and G, respectively. Table 5 illustrates that the minimum caching process and forwarding operations have been minimized when the hit occurs.

Table 5. Caching process at t = 9 to t = 12.

Time **t**	**RC** $\mathbf{s}_i$	**RIP** $\text{Ř}_{s_{ip}}$	**LU** Ĺ_s	**FIP** $\text{Ḟ}_{s_{ip}}$	**REDP** $\text{Ȑ}_{s_{edp}}$	**RDP** $\text{Ŕ}_{s_{dp}}$	**CC** Ḉ_s	**EC** È_s
9	G	$\text{Ř}_{G_{ip}}$ **(1)**	Ĺ_G **(2)**	$\text{Ḟ}_{G_{ip}}$ **(3)**	$\text{Ȑ}_{G_{edp}}$ **(4)**	$\text{Ŕ}_{G_{dp}}$ **(5)**	Ḉ_G **(7)**	È_C **(6)**
10	H	$\text{Ř}_{H_{ip}}$ **(1)**	Ĺ_H **(2)**	$\text{Ḟ}_{H_{ip}}$ **(3)**	$\text{Ȑ}_{H_{edp}}$ **(4)**	$\text{Ŕ}_{H_{dp}}$ **(5)**	Ḉ_H **(7)**	È_A **(6)**
11	E	$\text{Ř}_{E_{ip}}$ **(1)**	Ĺ_E **(2)**		$\text{Ȑ}_{E_{edp}}$ **(3)**			
12	G	$\text{Ř}_{G_{ip}}$ **(1)**	Ĺ_G **(2)**		$\text{Ȑ}_{G_{edp}}$ **(3)**			

Table 6. IMU process at t = 9 to t = 12.

Time	$\mathbf{s}_i$	$\text{Ꞇ}_{cŕ,s_i}$	$\text{Ƒ}_{č,s_i}$	$\text{Ţ}_{p,s_i}$	$\text{M}_{cÏ,s_i}$	**Time**	$\mathbf{s}_i$	$\text{Ꞇ}_{cŕ,s_i}$	$\text{Ƒ}_{č,s_i}$	$\text{Ţ}_{p,s_i}$	$\text{M}_{cÏ,s_i}$
t = 9	B	6	1	7	0.14	t = 11	E	11	2	2	1.00
	A	4	1	9	0.11		B	6	1	7	0.14
	C	3	1	10	0.10		D	5	1	8	0.13
	D	5	1	8	0.13		F	8	1	5	0.20
	E	7	1	6	0.17		G	9	1	4	0.25
	F	8	1	5	0.20		H	10	1	3	0.33
	G	9	1	4	0.25	t = 12	G	12	2	1	2.00
t = 10	B	6	1	7	0.14		E	11	2	2	1.00
	A	4	1	9	0.11		B	6	1	7	0.14
	D	5	1	8	0.13		D	5	1	8	0.13
	E	7	1	6	0.17		F	8	1	5	0.20
	F	8	1	5	0.20		H	10	1	3	0.33
	G	9	1	4	0.25						
	H	10	1	3	0.33						

Table 7 describes the detailed process of caching at t = 13. The cache space CS is full, and the time window W_T has expired; Table 8 demonstrates that when the new time window starts, all the values of $\text{Ƒ}_{č,s_i}$ become one (1) and retain the values of $\text{Ꞇ}_{cŕ,s_i}$. The exact process that was performed at t = 9 and t = 10 is repeated at t = 13. The IMU used the $\text{Ꞇ}_{cŕ,s_i}$ and $\text{Ƒ}_{č,s_i}$ for calculating the maturity index $\text{M}_{cÏ,s_i}$ of the content s_i. This value indicates the maturity of the content with the specific time window W_T .

The tables demonstrate that the lower value of a content maturity index $\text{M}_{cÏ,s_i}$ represents a longer stay in the cache space, with a lower frequency (popularity) over a particular time frame W_T . Therefore, this content is evicted from the cache when the cache space is full. It takes some time to define the maturity/immaturity of new cached content. Therefore, the content should not be evicted without checking the level of a content maturity index; the tables indicate that the maturity index value of new cached content is greater than others. Content that has become popular over time, but loses its popularity, has a higher frequency than other content. Therefore, this kind of content stays in CS for a long time and wastes cache space. However, the window W_T is used to equalize the frequency of all contents after a specific time, and immature content is selected from the maturity index $\text{M}_{cÏ,s_i}$ to evict content from the CS. The proposed policy has significantly improved the cache hit ratio, bandwidth usage, latency, and path stretch.

Table 7. Caching process at t = 13.

Time **t**	**RC** $\mathbf{s}_i$	**RIP** $\check{\mathbf{R}}_{s_{ip}}$	**LU** $\acute{\mathbf{L}}_s$	**FIP** $\dot{\mathrm{F}}_{s_{ip}}$	**REDP** $\grave{\grave{\mathrm{R}}}_{s_{edp}}$	**RDP** $\acute{\mathbf{R}}_{s_{dp}}$	**CC** $\acute{\c{C}}_s$	**EC** $\grave{\mathrm{E}}_s$
13	I	$\check{\mathrm{R}}_{I_{ip}}$ (1)	$\acute{\mathrm{L}}_I$ (2)	$\dot{\mathrm{F}}_{I_{ip}}$ (3)	$\grave{\grave{\mathrm{R}}}_{I_{edp}}$ (4)	$\acute{\mathrm{R}}_{I_{dp}}$ (5)	$\acute{\c{C}}_I$ (7)	$\grave{\mathrm{E}}_D$ (6)

Table 8. IMU process at t = 13.

Time	$\mathbf{s}_i$	$Ƭ_{c\acute{r},s_i}$	$Ƒ_{\check{c},s_i}$	$Ţ_{p,s_i}$	$M_{c\ddot{I},s_i}$	**Time**	$\mathbf{s}_i$	$Ƭ_{c\acute{r},s_i}$	$Ƒ_{\check{c},s_i}$	$Ţ_{p,s_i}$	$M_{c\ddot{I},s_i}$
t = 13	G	12	1	5	0.20		F	8	1	9	0.11
	E	11	1	6	0.17		H	10	1	7	0.14
	B	6	1	11	0.10		I	13	1	4	0.25
	D	5	1	12	0.08						

4. Performance Evaluation

We performed a simulation in the GEANT network topology using the Icarus [13] simulator, to evaluate the performance of our policy. The GEANT topology consists of 40 nodes and 60 edges. The cache capacity of each node in the network is the same and ranges between 4% to 20% of the total content population. We used warm-up requests to settle caches before running the actual experiment, to minimize experimental errors. The cache warm-up requests are 40,000 and measured requests are also 40,000. We also used measured requests for performance evaluation. Zipf's law is used to distribute the popularity of the content and popularity distribution of the exponent alpha (α) $\in$ [0.6, 0.8, 1.0] used in our simulation. For fair comparison with state-of-the-art replacement policies, the popularity of requested contents follows a Zipf distribution with a parameter ranging from 0.6 to 1.0, as presented in [10]. The lower and higher values indicate a low and high correlation between content requests [30]. The parameters of our simulation setup are mentioned in Table 9.

Table 9. Simulation Parameters.

Parameters	**Value**
Warm-up Requests	40,000
Measured Requests	40,000
Model of Popularity	0.6, 0.8, 1.0
Total Contents	100,000
Cache Capacity	4–20%
Consumer Request Rate	1.0 request/s
Placement Policy	LCE, CL4M, ProbCache, LCD, opt-Cache
Topology	GEANT

The obtained results have been compared with state-of-the-art content replacement policies, including LRU, LFU, FIFO, and LFRU. To check the effectiveness of our approach, we compared popular cache placement policies, including Leave Copy Everywhere (LCE) [27], Cache Less for More (CL4M) [31], ProbCache [32], Leave Copy Down (LCD) [33], and opt-Cache [10], with our proposed replacement policy (IMU). These placement policies indicate the more redundant data to less redundant data in the network, respectively [10]. These placement policies indicate the more redundant data to less redundant data in the network. These results prove the effectiveness of our proposed technique with different cache sizes and populations, using various performance metrics such as cache hit ratio, latency, link load, and path stretch. These performance metrics are compared one by one, as explained below.

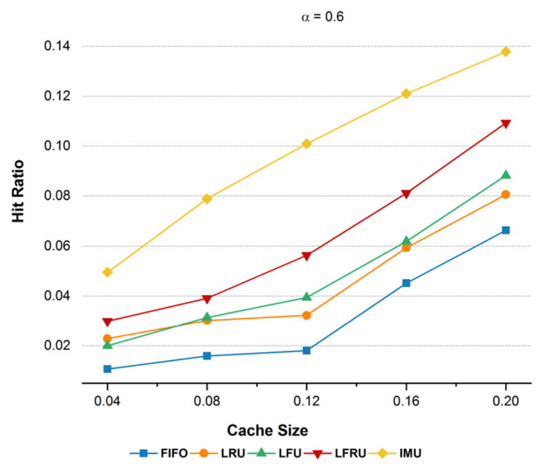
4.1. Cache Hit Ratio

The cache hit ratio is an essential metric for evaluating the performance of CCN cache. It identifies the response to network cache storage, in which content is cached locally within a specific time frame. Two terms are important in the cache hit ratio. The first is the cache hit (requested content is found from the cache), and the second is the cache miss (unlike cache hit). When content is available in the cache, the content request does not forward to the publisher. Therefore, a higher hit ratio indicates good cache performance and represents low bandwidth utilization, reduction in latency, and low server load. The cache hit ratio is defined as follows:

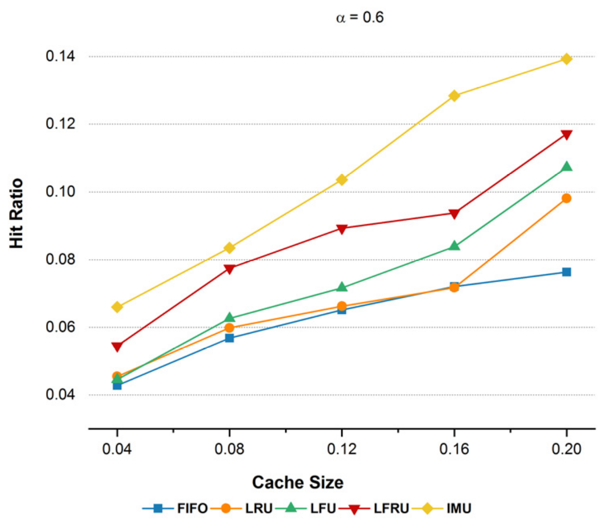
$$Hit\ Ratio = \frac{Cache_{hits}}{Cache_{hits} + Server_{hits}} \tag{1}$$

Our proposed strategy, IMU, compared to existing well-known replacement strategies in terms of the cache hit ratio. We have extracted the results from low to high popularity and different cache sizes. We first comment that content eviction policies behave the same under different caching strategies. Regardless of the content eviction policy, we observe in Figure 2 that the opt_cache performs best and the LCE performs the worst in terms of the cache hit ratio. Moreover, different eviction policies affect the performance of the cache hit ratio.

Figure 2 illustrates that the IMU's performance is better than the existing replacement strategies; this is because the IMU not only considers the time $\text{Ƭ}_{c\acute{r},s_i}$ but also the frequency $\text{Ƒ}_{\check{c},s_i}$ of the requested content within the specific period W_T . When the W_T is expired, then all the $\text{Ƒ}_{\check{c},s_i}$ initialize to their starting frequency ($\text{Ƒ}_{\check{c},s_i}$= 1). Moreover, it helps to evict content from the cache space whose popularity increases for a while and decreases shortly. When the cache is full, it is evicted from the cache after selecting the least value of the maturity index $\text{Ϻ}_{c\ddot{I},s_i}$. The advantage of immature content eviction from the cache is that most of the content is mature, which leads to a higher cache hit ratio.

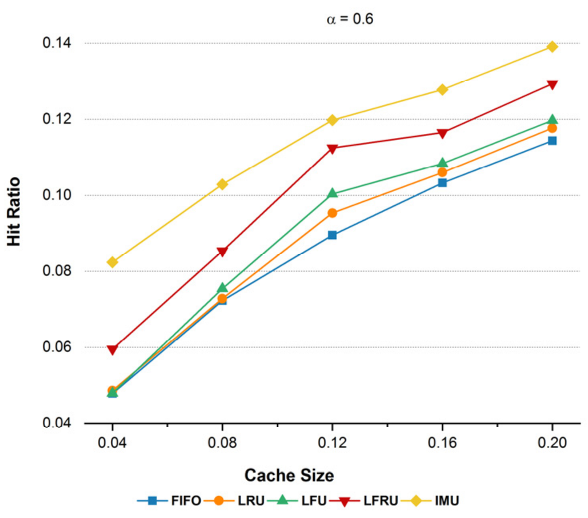


(**a**)

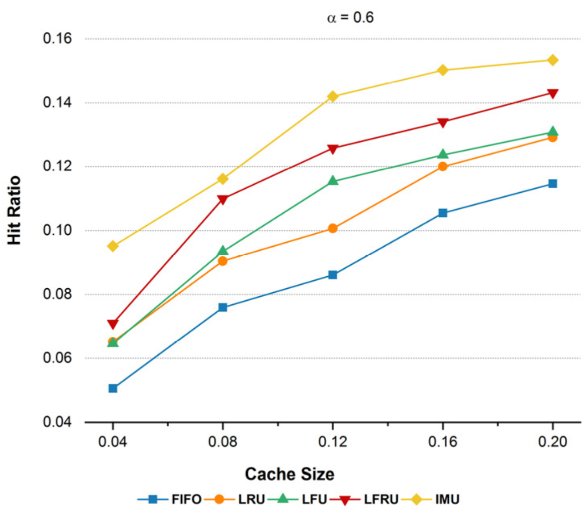


(**b**)

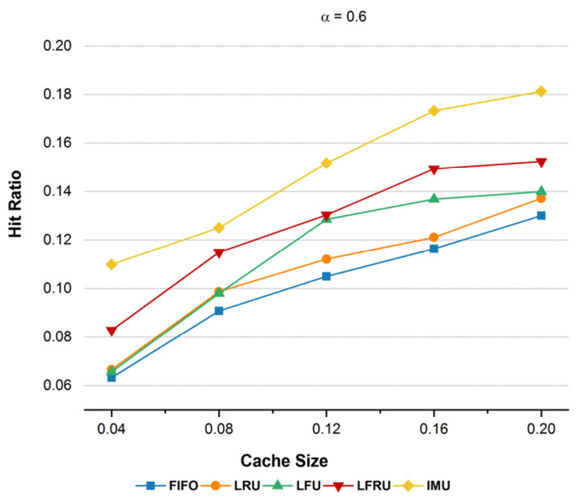
Figure 2. *Cont.*



(**c**)



(**d**)



(**e**)

Figure 2. Caching hit ratio with different cache sizes and α by using different placement policies. (**a**) Cache hit ratio with LCE. (**b**) Cache hit ratio with CL4M. (**c**) Cache hit ratio with ProbCache. (**d**) Cache hit ratio with LCD. (**e**) Cache hit ratio with opt-Cache.

We observed that FIFO underperformed because contents are removed from the cache in the same order in which they were cached, regardless of how many times they were previously accessed. Besides, increasing cache space and similar content requests improve FIFO's performance because the content stays in the cache for a longer period, which increases the chances of increasing the cache hit ratio. LFU performs better than LRU when the cache size is large and the content is repeatedly requested because LFU considers the frequency of the requested content, while LRU does not. Moreover, LFU caches popular content and evicts unpopular content from the cache. Besides, contents are often evicted from the cache when the cache size is small. However, LFU displays low performance in small cache sizes. LFRU has better performance due to the coupling of LRU and LFU; however, when the content request rate is minimum from the maximum normalized request rate, the content is evicted from the unprivileged partition. Therefore, the new content is cached in the unprivileged partition. Besides, if the content request rate is higher than the maximum normalized request rate, it chooses the least recent content from the privileged partition and pushes that content into the unprivileged partition. Hence, new content is cached in the privileged partition and hit counter associated with each partition. However, content that loses popularity stays in the unprivileged partition for a long time due to its high frequency. IMU outperformed FIFO, LRU, LFU, and LFRU in terms of the cache hit ratio by 48.33%, 30.07%, 26.34%, and 14.31%, respectively.

The percentage (%) of IMU performance in different popularities, and low to high cache sizes with different content placement strategies, is presented in Table 10. We observed that IMU is outperformed with low popularity because, if such content is popular for some time but its popularity decreases with time and its frequency is high, then IMU evicts this content from the cache space. When the cache space is low and the popularity of the content changes frequently, it becomes very difficult for the content eviction policy to decide which content should be removed from the cache space. Hence, the IMU policy evicts immature content from the cache space and gives each content an equal opportunity to define its maturity/immaturity level. Such content is not removed from the cache space that is gaining popularity.

4.2. Path Stretch (Hop Count)

Path stretch indicates the distance traveled to the content provider by the consumer's interest. The value of the path stretch is low when the consumer's interest packet is found from the routing path. Therefore, the better content replacement policy identifies content that users are interested in and that is mature. Such content should not be evicted from the cache. If such content is evicted from the cache, the publisher's load and bandwidth utilization will be high. Therefore, a better content replacement strategy should be to minimize the hops between the consumer and the publisher. Path stretch is defined as follows:

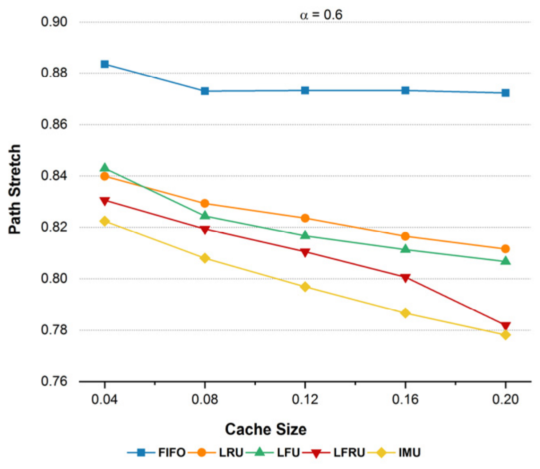
$$Path\ Stretch = \frac{\sum_{i=1}^{n} Hop - Traveled}{\sum_{i=1}^{n} THop - Hop} \tag{2}$$

where $\sum_{i=1}^{n} Hop - Traveled$ is the number of hops between the consumer and publisher nodes covered by consumer interest. The value $\sum_{i=1}^{n} THop - Hop$ denotes the total number of hops between the consumer and the provider. n represent the total number of generated interests for specific content.

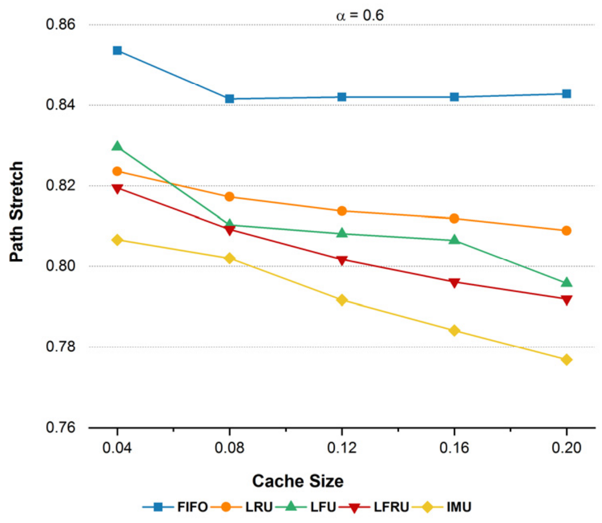
Table 10. IMU cache hit ratio percentage improvement.

α	Cache Size	Placement	FIFO (%)	LRU (%)	LFU (%)	LFRU (%)
0.6		LCE	320.35	133.19	121.34	64.43
0.6		CL4M	64.17	52.41	41.74	20.14
0.6	4% to 20%	ProbCache	38.71	35.01	32.35	16.53
0.6		LCD	56.49	31.90	26.71	14.37
0.6		opt-Cache	48.84	40.46	33.82	18.57
0.8		LCE	47.34	36.94	27.08	17.86
0.8		CL4M	27.84	25.22	18.72	12.93
0.8	4% to 20%	ProbCache	17.64	14.11	11.70	6.38
0.8		LCD	20.15	14.08	13.66	6.53
0.8		opt-Cache	22.30	17.98	17.12	9.48
1.0		LCE	25.60	22.04	21.29	14.05
1.0		CL4M	11.47	9.76	9.24	5.35
1.0	4% to 20%	ProbCache	7.56	6.21	6.38	1.98
1.0		LCD	7.49	4.96	6.82	2.59
1.0		opt-Cache	9.02	6.74	7.21	3.46

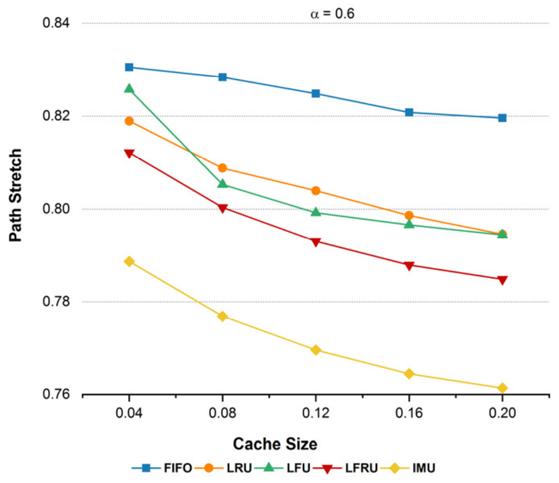
Figure 3 illustrates that the IMU's performance in terms of path stretch is better than other existing replacement policies. The placement strategy chooses the location of the cache, which may reduce the number of hops. IMU removes content that has been in the cache for a long time but has not matured. Therefore, when immature content is removed from the cache and new content is cached so that the consumer's requested content is available nearby, the request is not forward to the publisher. However, cached content on nearby routers is mostly popular or close to being popular.



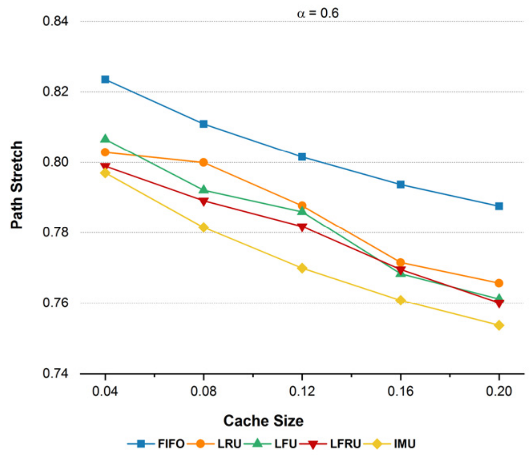
(**a**)



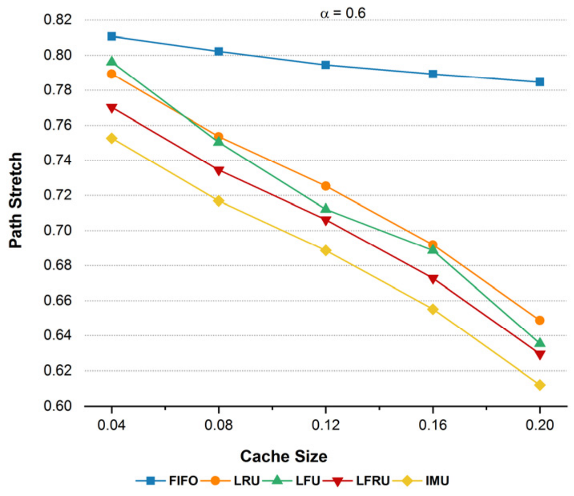
(**b**)



(**c**)



(**d**)

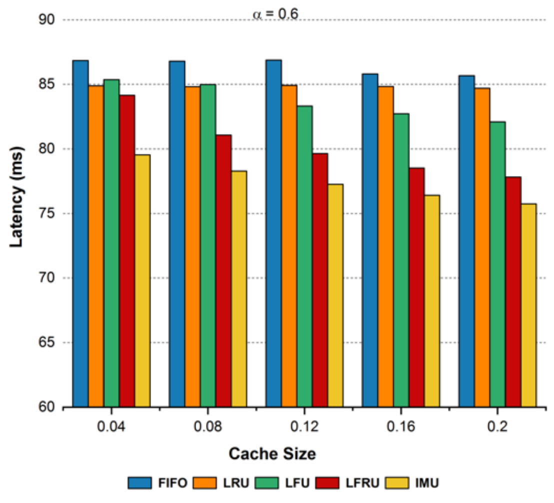


(**e**)

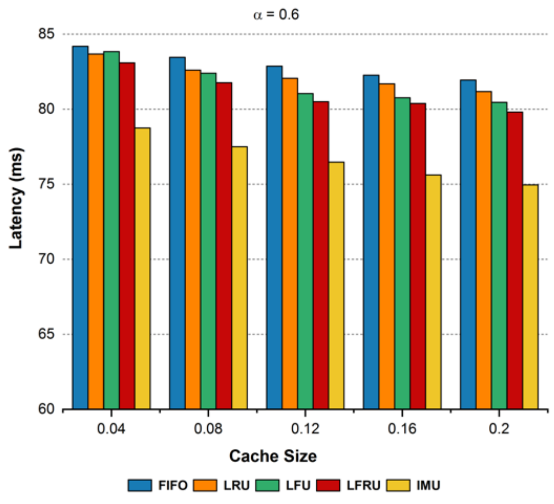
Figure 3. Path stretch with different cache sizes and α, using different placement policies. (**a**) Path stretch with LCE. (**b**) Path stretch with CL4M. (**c**) Path stretch with ProbCache. (**d**) Path stretch with LCD. (**e**) Path stretch with opt-Cache.

FIFO, LRU, and LFU represent the high path stretch due to content selection for eviction based on a single factor. FIFO content is evicted in the order in which it was cached. However, no matter how many times the content has been accessed, the timeline of popular and unpopular contents in FIFO will be the same, which increases the path stretch value. Figure 4 indicates that LRU is better than LFU when the cache size is smaller; however, as the cache size increases, the performance of LFU improves because LFU considers the popularity of content. Therefore, as cache size increases, popular content stays longer in the cache. LRU ignores the popularity of the content and the least recently used content evicts from the cached. However, content that is not popular, but, over time, their request keeps coming, are present in the cache space, making the path stretch higher. LFRU divides the cache space into two parts: LRU used privilege partition and LFU used unprivileged partition. With the higher request rate, the least recently used content has been evicted from the privilege partition and that content pushes it to the unprivileged partition. When unpopular content is pushed into the unprivileged partition, the content stays in the cache space for a long time. Further, these techniques are not focused on the maturity of the content. IMU outperformed FIFO, LRU, LFU, and LFRU in terms of path stretch by 11.33%, 6.16%, 5.77%, and 3.82%, respectively.

Table 11 illustrates the improvement of IMU in terms of path stretch using different content placement strategies with content eviction policies. We have observed IMU perform better in low to high cache space. In addition, IMU is better in high popularity. When the cache space is full, IMU selects immature content and evicts it from the cache. Therefore, popular content and content that may be popular remain in the cache. However, the consumer's request for specific content is fulfilled from the nearest node.

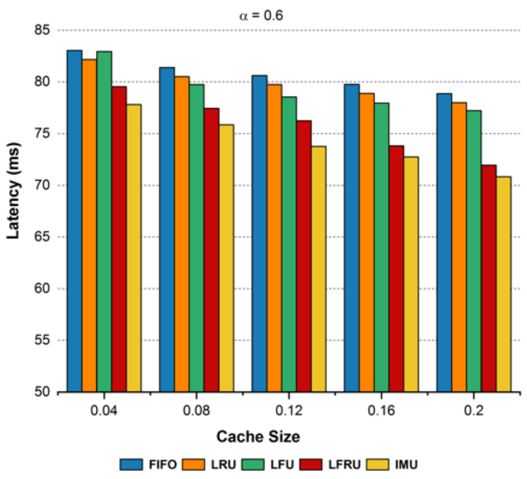


(**a**)

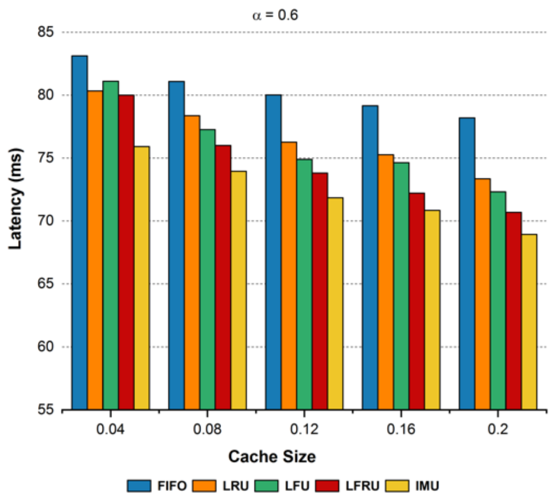


(**b**)

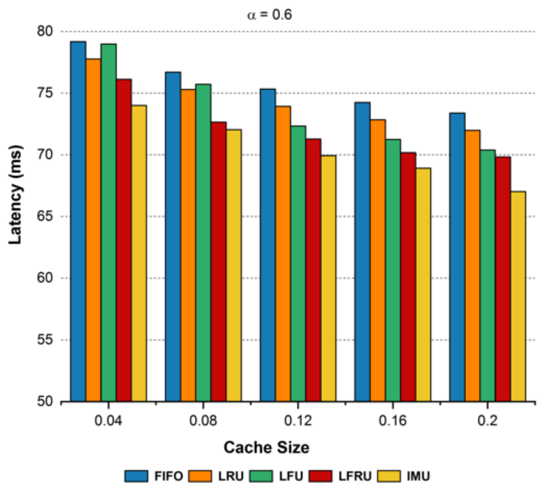
Figure 4. *Cont.*



(**c**)



(**d**)



(**e**)

Figure 4. Latency with different cache sizes and α, using different placement policies. (**a**) Latency with LCE. (**b**) Latency with CL4M. (**c**) Latency with ProbCache. (**d**) Latency with LCD. (**e**) Latency with opt-Cache.

Table 11. IMU path stretch percentage improvement.

α	Cache Size	Placement	FIFO (%)	LRU (%)	LFU (%)	LFRU (%)
0.6	4% to 20%	LCE	8.76	3.97	2.68	1.26
0.6		CL4M	6.17	2.80	2.20	1.42
0.6		ProbCache	6.38	4.07	3.98	2.95
0.6		LCD	3.84	1.65	1.30	0.94
0.6		opt-Cache	14.02	5.10	4.35	2.51
0.8	4% to 20%	LCE	10.32	4.88	4.30	2.44
0.8		CL4M	8.56	5.40	5.34	3.14
0.8		ProbCache	9.43	4.20	3.62	3.05
0.8		LCD	7.42	4.96	4.20	3.16
0.8		opt-Cache	10.14	6.10	5.56	3.18
1.0	4% to 20%	LCE	21.51	14.47	14.54	10.00
1.0		CL4M	20.35	9.04	9.27	6.25
1.0		ProbCache	17.03	8.16	8.30	5.25
1.0		LCD	10.73	8.49	8.69	7.13
1.0		opt-Cache	15.30	9.06	8.17	4.68

4.3. Latency

Latency indicates the delay in the delivery of requests and content from consumers. It is a vital metric for evaluating the performance of the CCN cache, and it is defined as follows:

$$Latency = Request\ Travel\ Delay + Content\ Travel\ Delay \tag{3}$$

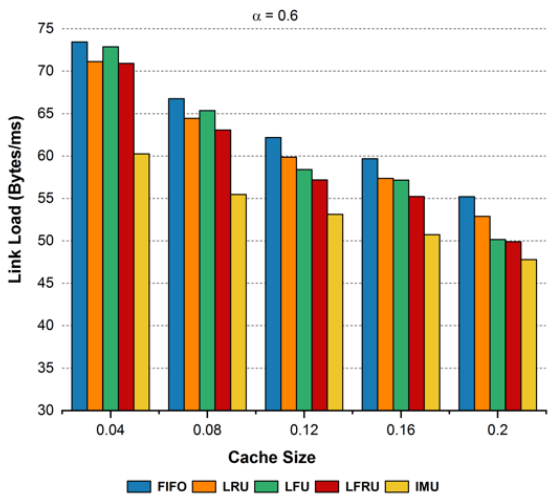
The IMU provides low latency because it evicts the most suitable content from the cache, based on immaturity. If the cache is full, the IMU jointly considers the frequency and time and selects the content for potential eviction from the cache. Hence, more popular and mature content will be in the cache, and content that may be popular. However, most consumer requests are satisfied along the routing path, which reduces latency. Figure 5 illustrates that IMU's performance is better than other content replacement policies, regarding latency with different cache sizes and popularity.

Figure 5 illustrates that FIFO represents a high latency because the duration of popular and unpopular content is the same. However, latency increases when popular content is evicted from the cache. LRU ignores the popularity of the content. Therefore, requests for less popular content come before eviction, and the content will remain in the cache, which causes high latency. LFU considers the frequency of the content, and contents that increase in frequency over a short period of time but are no longer popular; such contents use cache space due to their high frequency.

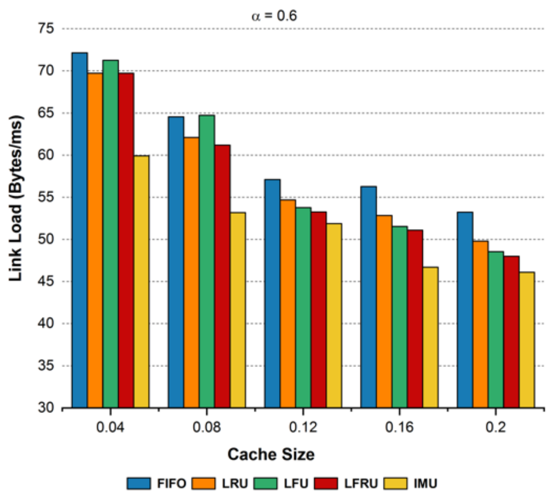
Therefore, fresh contents are reduced in the cache, which increases the latency. LFRU performs better than the previous two discussed replacement techniques because LRU and LFU are used together. When the request rate is high, then the required processing should be high, because the least recently used content is evicted from the privileged partition and pushed to the unprivileged partition and associated with the access history of content. In addition, low-frequency content is evicted from unprivileged partition. However, content with a high access history that is no longer popular will spend more time in the cache space, reducing the freshness of the content. IMU outperformed FIFO, LRU, LFU, and LFRU in terms of latency by 12.32%, 9.97%, 9.08%, and 5.91%, respectively.

When the alpha equals 0.8 with cache size is 0.04, IMU is 64.44 ms, which is 9.45% lower than LFRU (71.16 ms), 13.90% lower than LFU (74.84 ms), 13.34% lower than LRU (74.35 ms), and 15.60% lower than FIFO (76.34 ms). When the alpha equals 0.8, the cache size is 0.12, IMU is 61.03 ms, which is 5.40% lower than LFRU (64.51 ms), 9.65% lower than LFU (67.55 ms), 12.05% lower than LRU (69.39 ms), and 14.50% lower than FIFO (71.38 ms). When the alpha equals 0.8 with cache size is 0.2, IMU is 60.05 ms, which is 3.71% lower than LFRU (62.37 ms), 5.58% lower than LFU (63.60 ms), 8.93% lower than LRU (65.94 ms), and 11.59% lower than FIFO (67.92 ms). Therefore, as the cache size increases, the latency is reduced because more content in the network can be cached.

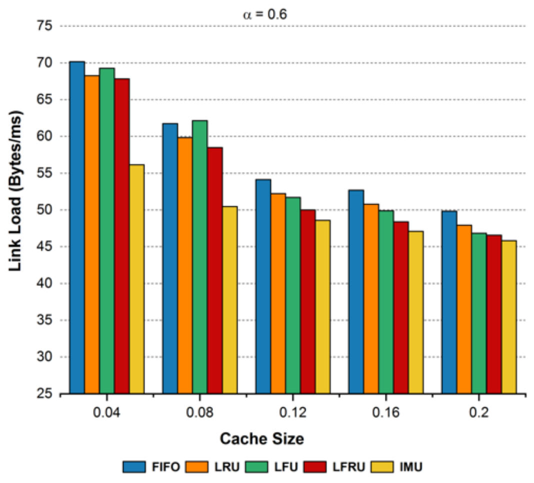
The latency improvement performance from IMU is illustrated in Table 12, using different content placement strategies with low to high popularities and cache sizes. We have observed that, as the popularity of content increases, so performs IMU. The IMU evicts content from the cache that has been in the cache for a long time and has few requests. Furthermore, content that has been in high demand for some time but declined over time has also been evicted from the cache. Therefore, the cache contains mostly mature content. However, when a consumer requests specific content, the consumer's request does not reach the publisher because the consumer is satisfied along the routing path.



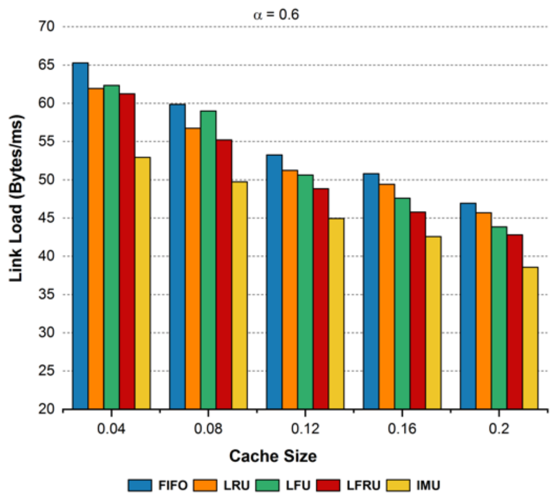
(**a**)



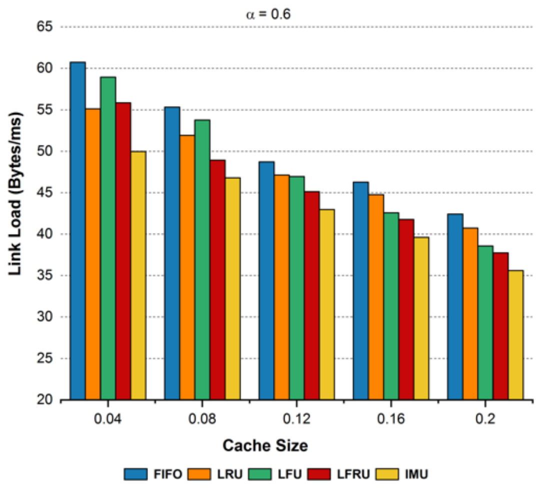
(**b**)



(**c**)



(**d**)



(**e**)

Figure 5. Link load with different cache sizes and α, using different placement policies. (**a**) Link load with LCE. (**b**) Link load with CL4M. (**c**) Link load with ProbCache. (**d**) Link load with LCD. (**e**) Link load with opt-Cache.

Table 12. IMU latency percentage improvement.

α	Cache Size	Placement	FIFO (%)	LRU (%)	LFU (%)	LFRU (%)
0.6		LCE	10.37	8.71	7.48	3.45
0.6		CL4M	7.58	6.79	6.17	5.49
0.6	4% to 20%	ProbCache	8.13	7.12	6.43	2.10
0.6		LCD	10.00	5.76	4.90	6.55
0.6		opt-Cache	7.12	5.37	4.51	2.26
0.8		LCE	14.01	11.55	10.08	6.08
0.8		CL4M	14.32	9.65	8.71	6.30
0.8	4% to 20%	ProbCache	7.39	6.12	5.44	1.95
0.8		LCD	10.12	7.01	6.36	4.07
0.8		opt-Cache	9.26	7.18	5.97	3.38
1.0		LCE	21.37	19.60	18.87	9.98
1.0		CL4M	18.36	15.96	15.24	13.53
1.0	4% to 20%	ProbCache	14.55	12.39	12.24	7.51
1.0		LCD	14.61	11.90	11.08	6.89
1.0		opt-Cache	17.67	14.39	12.74	9.05

4.4. Link Load

Link load indicates the total number of bytes (consumer's request size and content size) traversed for retrieving the interesting content at the specific time limit. It measures bandwidth usage in the network and is defined as follows:

$$Link\ Load = \frac{(request_{size} \times request_{link\_count}) + (content_{size} \times content_{link\_count})}{Duration} \tag{4}$$

$$Duration = Content\ Retrieval\ Time - Content\ Request\ Time \tag{5}$$

where, $request_{size}$ denotes the request's size in bytes, $request_{link\_count}$ designates the number of the links traversed that reach the source, $content_{size}$ is the content size to retrieve, $content_{link\_count}$ is the number of links where the content reaches the request's originator.

Figure 5 illustrates that IMU performs better than other existing strategies in terms of link load. IMU does not replace such content from the cache, which has a frequency over a certain period of time. Therefore, consumer request is mostly satisfied with the routing path or close to the consumer. Therefore, most of the content in the cache is of interest to the user. In addition, content that increases in frequency for some time but does not become popular later also removes such content from the cache. However, IMU maintains the freshness of the content as well as the mature content in the cache.

FIFO does not compete with the popularity of content because this technique only considers the order in which the content is cached and evicts the content from the cache in that order. Therefore, popular content is evicted from the cache. However, most consumer requests are satisfied with the publisher. Figure 5 demonstrates that LRU is better than LFU when the cache size is smaller. LFU performance improves as the cache size increases, as LFU takes into account the popularity of the content. Therefore, popular content stays in the cache for a long time, and the consumer's request is found in the cache space and not forwarded to the publisher. In addition, content that increases in frequency stays in the cache space, even if it is not popular. However, this is a misuse of cache space and leads to a higher link load. LRU ignores the popularity of the content as well as the maturity of the content. Therefore, popular content requested in the past is likely to be used in the future, but recently requested content may be replaced with less popularity; thus, it does not adapt to changing workloads. When the request rate is high in LFRU, the least recently used content is evicted from the privileged partition and pushed towards the unprivileged partition, with complete access history. However, this content is no longer popular but has a high access history; this content spends more time in cache space, which causes high link load. IMU outperformed FIFO, LRU, LFU, and LFRU in terms of link load by 18.04%, 13.61%, 12.49%, and 9.53%, respectively.

When the alpha equals 0.8, with a cache size of 0.04, IMU is 55.48 bytes/ms, which is 16.41% lower than LFRU (66.37 ms), 19.60% lower than LFU (69.01 bytes/ms), 17.85% lower than LRU (67.53 bytes/ms), and 20.57% lower than FIFO (69.85 bytes/ms). When the alpha equals 0.8, with a cache size is 0.12, IMU is 48.99 bytes/ms, which is 15.25% lower than LFRU (57.81 bytes/ms), 18.57% lower than LFU (60.16 bytes/ms), 19.21% lower than LRU (60.64 bytes/ms), and 22.19% lower than FIFO (62.96 bytes/ms). When the alpha equals 0.8, with cache size is 0.2, IMU is 44.93 bytes/ms, which is 10.40% lower than LFRU (50.15 bytes/ms), 10.68% lower than LFU (50.30 bytes/ms), 11.51% lower than LRU (50.78 bytes/ms), and 15.37% lower than FIFO (53.09 bytes/ms). As the cache size increases, we observed that the link load decreases, as the proposed scheme removes immature content from the cache. Therefore, IMU maintains the data freshness with popularity within the network. However, none of the previous eviction policies have adopted the concept of immaturity for content selection.

Table 13 describes the IMU's improvement in percentage (%) of the link load, which used different content placement strategies along with content eviction policies. We observed that IMU outperformed the other content eviction policies against low to high popularity and cache space. It performed better in a fully redundant and low redundancy environments. IMU contains the most popular and mature content in the cache and makes better use of cache space. Moreover, the consumer is mostly satisfied along the routing path when requesting content. Therefore, the link load value is low because the request is not sent to the publisher.

Table 13. IMU link load percentage improvement.

α	Cache Size	Placement	FIFO (%)	LRU (%)	LFU (%)	LFRU (%)
0.6	4% to 20%	LCE	15.58	12.34	11.49	9.30
0.6		CL4M	14.83	10.52	10.34	8.46
0.6		ProbCache	13.45	10.40	10.30	7.61
0.6		LCD	17.08	13.71	12.93	9.68
0.6		opt-Cache	15.07	10.43	10.27	6.10
0.8	4% to 20%	LCE	20.32	17.19	17.19	14.40
0.8		CL4M	21.54	18.04	17.45	15.28
0.8		ProbCache	19.18	16.17	14.95	12.04
0.8		LCD	23.44	18.48	16.51	14.64
0.8		opt-Cache	22.03	11.74	9.19	6.97
1.0	4% to 20%	LCE	20.89	17.40	15.27	13.09
1.0		CL4M	16.10	11.15	9.99	5.95
1.0		ProbCache	16.23	12.78	11.73	7.43
1.0		LCD	17.56	11.62	9.37	6.40
1.0		opt-Cache	17.30	12.14	10.35	5.55

5. Conclusions and Future Work

In-network caching is one of the essential features in the CCN architecture network, allowing content items to be cached in the router nodes for some time, to meet subsequent consumer requests. Due to the limited cache capacity in the node, any cached content in the cache needs to be evicted to accommodate new content. Content replacement policy is responsible for choosing the right content against defined criteria. Existing cache replacement policies use the concept of popularity or time for content eviction. However, when content loses its popularity after becoming very popular in a certain period, it remains in the cache space. Moreover, content is evicted from the cache space before it becomes popular. Therefore, the proposed policy handles cached items that lose their popularity over a specific time frame and remain in the cache for a long time. We introduced the new concept of content maturity and immaturity for content eviction in CCN. The proposed content replacement policy (IMU) uses the concept of maturity/immaturity of the content. This policy finds the content maturity index by using the content arrival time and its frequency. Also, it determines the maturity level through a maturity classifier. We have performed extensive simulations to evaluate the proposed content replacement policy, using the Icarus simulator under different cache sizes and content popularity. The simulation results indicate that the proposed policy outperformed recent and baseline content replacement policies (FIFO, LRU, LFU, and LFRU). The results demonstrate that the proposed policy is better in terms of the cache hit ratio, latency, path stretch, and link load. In the future, this work can be extended to use the content replacement policy (IMU) with different constraints in different use cases. Another potential future work is in-depth investigation of content diversity for the nodes with very high content popularity.

Author Contributions: Conceptualization, S.R., S.A.R. and F.A.G.; methodology, S.R.; software, S.R.; validation, S.R., S.A.R. and F.A.G.; formal analysis, S.R. and F.A.G.; investigation, S.R., S.A.R. and F.A.G.; writing—original draft preparation, S.R.; writing—review and editing, S.R., S.A.R. and F.A.G.; supervision, S.A.R. and F.A.G.; funding acquisition, S.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Universiti Teknologi Malaysia (UTM) under UTM Research University Grant Scheme: (VOT Q.J130000.3613.03M41), Universiti Teknologi Malaysia, Johor Bahru, Malaysia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gui, Y.; Chen, Y. A Cache Placement Strategy Based on Compound Popularity in Named Data Networking. *IEEE Access* **2020**, *8*, 196002–196012. [CrossRef]
2. Khandaker, F.; Oteafy, S.; Hassanein, H.S.; Farahat, H. A functional taxonomy of caching schemes: Towards guided designs in information-centric networks. *Comput. Netw.* **2019**, *165*, 106937. [CrossRef]
3. Noh, H.; Song, H. Progressive Caching System for Video Streaming Services Over Content Centric Network. *IEEE Access* **2019**, *7*, 47079–47089. [CrossRef]
4. Zheng, X.; Wang, G.; Zhao, Q. A Cache Placement Strategy with Energy Consumption Optimization in Information-Centric Networking. *Futur. Internet* **2019**, *11*, 64. [CrossRef]
5. Ioannou, A.; Weber, S. A survey of caching policies and forwarding mechanisms in information-centric networking. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2847–2886. [CrossRef]
6. Ullah, R.; Rehman, M.A.U.; Naeem, M.A.; Kim, B.-S.; Mastorakis, S. ICN with edge for 5G: Exploiting in-network caching in ICN-based edge computing for 5G networks. *Futur. Gener. Comput. Syst.* **2020**, *111*, 159–174. [CrossRef]
7. Naeem, M.A.; Ali, R.; Alazab, M.; Yhui, M.; Zikria, Y. Bin Enabling the content dissemination through caching in the state-of-the-art sustainable information and communication technologies. *Sustain. Cities Soc.* **2020**, *61*, 102291. [CrossRef]
8. Naeem, M.A.; Rehman, M.A.U.; Ullah, R.; Kim, B.-S. A Comparative Performance Analysis of Popularity-Based Caching Strategies in Named Data Networking. *IEEE Access* **2020**, *8*, 50057–50077. [CrossRef]
9. Ji, Y.; Zhang, X.; Liu, W.; Zhang, G. Replacement based content popularity and cache gain for 6G content-centric network. *Phys. Commun.* **2021**, *44*, 101238. [CrossRef]
10. Qazi, F.; Khalid, O.; Rais, R.N.B.; Khan, I.A. Optimal Content Caching in Content-Centric Networks. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 6373960. [CrossRef]
11. Khattak, H.; Ul Amin, N.; Din, I.U.; Insafullah; Iqbal, J. LeafPopDown: Leaf Popular Down Caching Strategy for Information-Centric Networking. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 148–151. [CrossRef]
12. Kumar, S.; Tiwari, R. Dynamic popularity window and distance-based efficient caching for fast content delivery applications in CCN. *Eng. Sci. Technol. Int. J.* **2021**, *24*, 829–837. [CrossRef]
13. Liu, H.; Han, R. A Hierarchical Cache Size Allocation Scheme Based on Content Dissemination in Information-Centric Networks. *Futur. Internet* **2021**, *13*, 131. [CrossRef]
14. Amadeo, M.; Ruggeri, G.; Campolo, C.; Molinaro, A. Diversity-improved caching of popular transient contents in vehicular named data networking. *Comput. Netw.* **2021**, *184*, 107625. [CrossRef]

15. Gui, Y.; Chen, Y. A Cache Placement Strategy Based on Entropy Weighting Method and TOPSIS in Named Data Networking. *IEEE Access* **2021**, *9*, 56240–56252. [CrossRef]
16. Kalghoum, A.; Gammar, S.M.; Saidane, L.A. Towards a novel cache replacement strategy for Named Data Networking based on Software Defined Networking. *Comput. Electr. Eng.* **2018**, *66*, 98–113. [CrossRef]
17. Seyyed Hashemi, S.N.; Bohlooli, A. Analytical characterization of cache replacement policy impact on content delivery time in information-centric networks. *Int. J. Commun. Syst.* **2019**, *32*, e4154. [CrossRef]
18. Ai, L.; Deng, Y.; Zhou, Y.; Feng, H. RUE: A caching method for identifying and managing hot data by leveraging resource utilization efficiency. *Softw. Pract. Exp.* **2021**. [CrossRef]
19. Saino, L.; Psaras, I.; Pavlou, G. Icarus: A caching simulator for information centric networking (icn). In Proceedings of the SimuTools, Lisbon, Portugal, 17–19 March 2014; ICST: Ghent, Belgium, 2014; Volume 7, pp. 66–75.
20. Goian, H.S.; Al-Jarrah, O.Y.; Muhaidat, S.; Al-Hammadi, Y.; Yoo, P.; Dianati, M. Popularity-based Video Caching Techniques for Cache-enabled Networks: A survey. *IEEE Access* **2019**, *7*, 27699–27719. [CrossRef]
21. Pfender, J.; Valera, A.; Seah, W.K.G. Performance comparison of caching strategies for information-centric IoT. In Proceedings of the Proceedings of the 5th ACM Conference on Information-Centric Networking, Boston, MA, USA, 21–23 September 2018; ACM: New York, NY, USA, 2018; pp. 43–53.
22. Paschos, G.S.; Iosifidis, G.; Tao, M.; Towsley, D.; Caire, G. The role of caching in future communication systems and networks. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1111–1125. [CrossRef]
23. Chen, L.; Song, L.; Chakareski, J.; Xu, J. Collaborative Content Placement among Wireless Edge Caching Stations with Time-to-Live Cache. *IEEE Trans. Multimed.* **2019**, *22*, 432–444. [CrossRef]
24. Karakostas, G.; Serpanos, D.N. Exploitation of different types of locality for web caches. In Proceedings of the ISCC 2002 Seventh International Symposium on Computers and Communications, Taormina-Giardini Naxos, Italy, 1–4 July 2002; IEEE: Piscataway, NJ, USA, 2002; pp. 207–212.
25. Bilal, M.; Kang, S.-G. A cache management scheme for efficient content eviction and replication in cache networks. *IEEE Access* **2017**, *5*, 1692–1701. [CrossRef]
26. Zhang, Y.; Tan, X.; Li, W. PPC: Popularity prediction caching in ICN. *IEEE Commun. Lett.* **2017**, *22*, 5–8. [CrossRef]
27. Meddeb, M.; Dhraief, A.; Belghith, A.; Monteil, T.; Drira, K.; Mathkour, H. Least fresh first cache replacement policy for NDN-based IoT networks. *Pervasive Mob. Comput.* **2019**, *52*, 60–70. [CrossRef]
28. Khan, J.; Westphal, C.; Garcia-Luna-Aceves, J.; Ghamri-Doudane, Y. Nice: Network-oriented information-centric centrality for efficiency in cache management. In Proceedings of the 5th ACM Conference on Information-Centric Networking, Boston, MA, USA, 21–23 September 2018; ACM: New York, NY, USA, 2018; pp. 31–42.
29. Lal, K.N.; Kumar, A. A popularity based content eviction scheme via betweenness-centrality caching approach for content-centric networking (CCN). *Wirel. Netw.* **2019**, *25*, 585–596. [CrossRef]
30. Kumar, S.; Tiwari, R. An efficient content placement scheme based on normalized node degree in content centric networking. *Cluster Comput.* **2021**, *24*, 1277–1291. [CrossRef]
31. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache "less for more" in information-centric networks. In Proceedings of the International Conference on Research in Networking, Prague, Czech Republic, 21–25 May 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 27–40.
32. Naeem, M.A.; Nor, S.A.; Hassan, S.; Kim, B.-S. Compound Popular Content Caching Strategy in Named Data Networking. *Electronics* **2019**, *8*, 771. [CrossRef]
33. Seetharam, A. On caching and routing in information-centric networks. *IEEE Commun. Mag.* **2017**, *56*, 204–209. [CrossRef]
34. Ren, J.; Qi, W.; Westphal, C.; Wang, J.; Lu, K.; Liu, S.; Wang, S. MAGIC: A distributed MAx-gain in-network caching strategy in information-centric networks. In Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 27 April–2 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 470–475.