future internet

MDPI

Article

Weighted-CAPIC Caching Algorithm for Priority Traffic in Named Data Network

Leanna Vidya Yovita [1,*], Nana Rachmana Syambas [2] and Ian Joseph Matheus Edward [2]

[1] School of Electrical Engineering, Telkom University, Bandung 40257, Indonesia
[2] School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung 40116, Indonesia; nana@stei.itb.ac.id (N.R.S.); ian@stei.itb.ac.id (I.J.M.E.)
* Correspondence: leanna@telkomuniversity.ac.id; Tel.: +62-22-7564108

Abstract: Today, the internet requires many additional mechanisms or protocols to support various ever-growing applications. As a future internet architecture candidate, the Named Data Network (NDN) offers a solution that naturally fulfills this need. One of the critical components in NDN is cache. Caching in NDN solves bandwidth usage, server load, and service time. Some research about caching has been conducted, but improvements can be made. In this research, we derived the utility function of multiclass content to obtain the relationship between the class's weight and cache hit ratio. Then, we formulated it into the Weighted-CAPIC caching algorithm. Our research shows that Weighted-CAPIC provides a higher cache hit ratio for the priority class and the whole system. This performance is supported while the algorithm still provides the same path-stretch value as Dynamic-CAPIC. The Weighted-CAPIC is suitable to used in mobile nodes due to its ability to work individually without having to coordinate with other nodes.

Keywords: Named Data Network; cache; weight; class; priority

Citation: Yovita, L.V.; Syambas, N.R.; Edward, I.J.M. Weighted-CAPIC Caching Algorithm for Priority Traffic in Named Data Network. *Future Internet* **2022**, *14*, 84. https://doi.org/10.3390/fi14030084

Academic Editors: José Carlos Lopez-Ardao, Miguel Rodríguez Pérez and Sergio Herrería Alonso

Received: 20 February 2022
Accepted: 11 March 2022
Published: 12 March 2022

1. Introduction

Global internet traffic continues to increase every year. Based on Cisco's internet report in 2020, it is estimated that 67% of the world's population will be connected to the internet by 2023, meaning there will be 5.3 billion internet consumers. This number is an increase from a total of 3.9 billion consumers in 2018 [1]. Based on the Global Internet Phenomena Report 2019 by Sandvine, traffic related to streaming media is the highest, followed by user-generated content (UGC) and web access [2].

The current internet uses TCP/IP, a host-based network. In this architecture, consumer requests are addressed to a specific server address. This address is included in the request packet that the consumer sends. Only the destination server can respond to the consumer's request. The traffic growth cannot be supported efficiently by TCP/IP network. The network becomes burdened by the communication process.

Named Data Network (NDN) is a content-based architecture that focuses on the name of the content. Content names are used at the application level, and the other layer is used for caching, forwarding, routing, and other data processing [3]. The NDN is suitable for communication needs that are basically focused on content. In contrast to IP that cannot support this naturally, NDN makes communication run more efficiently. NDN is ready for a consumer character with dynamic demand. Another advantage is that we can treat traffic classes differently to give higher performance to the higher-priority classes based on the caching rules.

Caching as one of the primary and crucial components in NDN is a solution of bandwidth usage, server load, and service time [4]. Previous research has showed that algorithms with a different treatment for contents provide better performance. Kim et al. proposed treatment techniques for different service classes, both with forwarding techniques and caching techniques [5]. The service is divided into four classes, and there are different

policies for each class. Sourlas et al. proposed partition-based caching that gives separated space for contents and simulated some cases of partition with a particular forwarding algorithm [6]. Dehghan et al. proposed utility-driven caching that provides different allocation for content providers and content publishers based on the maximum utility [7]. Split-caching presented by Majd et al. used the technique to divide the content store into two parts, for priority and nonpriority content. The algorithm decides whether the content needs to be cached or not [8]. Yovita et al. proposed Dynamic-CAPIC, a caching algorithm that distinguishes the cache portion based on the popularity and class of the content in real-time [9]. This algorithm increases the network cache hit ratio (CHR) and provides performance that matches a particular class on the system. However, we need to develop a technique to give the best performance for the main priority class while increasing total system performance by efficiently using all available resources on the network, to provide a better experience for the consumer. In this research, we propose Weighted-Cache based on Popularity and Class (Weighted-CAPIC) as a solution to provide a proper performance for priority class. We derive the utility function formula for multiclass content to obtain the relationship between weight and cache hit ratio (CHR). This algorithm is suitable for the NDN mobile node because every router node can run this algorithm individually without coordinating with other nodes.

This research is written in the following order. Section 2 explains the basics of Named Data Networking. Section 3 describes the research related to caching algorithms on Named Data Network. Section 4 discusses our proposed caching algorithm, Weighted Cache based on Popularity and Class (Weighted-CAPIC) algorithm. Section 5 explains the system model, while Section 6 concludes the simulation results and analysis. Section 7 is about algorithm complexity. Section 8 is the conclusion, and Section 9 is the future research.

2. Named Data Network (NDN)

Named Data Network (NDN) is one of the candidates for future internet architecture because it offers a more efficient communication mechanism than the current internet (TCP/IP). The network must adaptively support various technological development and various consumer characteristics. Unlike NDN, which has these features naturally, the internet today (TCP/IP) requires many additional mechanisms or protocols to support multiple evolved applications [10]. IP networks add some mechanisms to maintain end-to-end connections and map the data with a specific address. On an IP network, the consumer sends a request for content to a particular server. This request message contains the address of the server. Therefore, the response to this consumer request is only made by a server with a specific address.

NDN has two types of packet: an interest packet containing a consumer's request and a data packet containing the information as a response to the interest packet. Every node on NDN has three components, namely Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Based (FIB) [11]. When an NDN router node receives an interest message containing a content request from customers, the node checks its content store (CS) for whether the node has the requested data in its cache. If the content exists in the router, it will be sent directly to the consumer. If not, then the router node will continue to check its PIT and update this request's information. The router will add information about the face from which the request came. The router also writes the requested content's name if this data does not exist before in the PIT. If there is no previous information about the requested content on the PIT, then the interest message will be forwarded to another node based on the information in the FIB. The information on FIB contains the next hop to reach the other node with content.

Content storage is a limited resource in NDN. Even if we can provide high memory due to the affordable price, using too large memory on the router node will cause high processing time and poor system performance. Therefore, it is essential to regulate the mechanism in the content store to obtain optimal performance.

3. Related Works: Caching Algorithms on Named Data Network

Caching is a fundamental feature available on NDN. In the IP (TCP/IP) networks, the buffer on the router also can store data, but after forwarding the data, it can no longer be used by the router [12]. This character causes a big difference between IP networks and NDN.

Until now, there have been many studies on NDN caching algorithms. The most common caching algorithm stored content on all nodes in the content delivery path. Regarding the content replacement in the content store, some basic algorithms commonly used in NDN are FIFO (first in, first out), Least Recently Used (LRU), and Least Frequently Used (LFU) [13]. FIFO stores content in the content store in the order it was entered and deletes the last order when the content store is full. The LRU removes content that has been unsolicited for a long time. LFU removes content that consumers rarely request. When the content store is full, it is necessary to select which content to delete so that the place can be used to store new content. According to research conducted by Jing et al., LFU is the most effective cache replacement strategy for networks with demand patterns following the Zipf–Mandelbrot distribution [14]. Hence we use LFU in this research.

Research on the content store's rule has been carried out previously, such as in utility-driven caching by Dehghan et al. [7]. In this scheme, content is modeled by generating it from different parties. Each party has a certain proportion in the NDN router's content store to keep the data based on their utility. The content is given a particular time to live. When the content has expired, it will be removed from the content store.

The other method was proposed by Kim et al. [5]. They presented the diff-caching model in NDN. There are four classes in this model, i.e., Dedicated Caching (DC), Assured Caching (AC), Best-Effort (BE), and Bypass Caching (BC). The caching process works by marking the data packet based on the service class. The data is always stored for the DC class content (up to a certain period). According to the producers' budget, the AC class content is stored within certain resource limits. The BE content class is stored if there are resources, and it should be ready to be deleted if the content store is full [5].

The caching algorithm in the previous studies mostly differentiates content treatment based on its popularity. The content store is partitioned to store popular and less popular content separately, as in the split-cache algorithm [8]. The split-caching proposed by Majd et al. selects the content to be cached based on the number of requests, and the number of hops traveled from the consumer to the router that stores the content [8].

The partition of content stores can also be done based on the content provider [15]. Sourlas et al. explained the partition-based caching that cached content based on its delivery rate [6]. The priority content is stored separately to avoid deleting priority content in the cache to be replaced with nonpriority content.

The partitions of the content store can be static, or they can be changed every time. One of the caching algorithms with static divisions in the content store is Static-CAPIC [16]. The authors classify traffic into three classes with different characters. In this algorithm, a different treatment has been carried out on each content class. Each class has a different portion of the storage. The Static-CAPIC algorithm has improved the cache hit ratio of the network. Still, it is deficient in providing performance for the priority class when the consumer's demand changes dynamically [9].

Yovita et al. developed this into Dynamic-CAPIC, where the cache portion for each content class can change according to demand conditions [9]. This increases the network cache hit ratio and provides a performance that matches the content class character in the system. However, we need to improve the performance of priority classes while increasing total system performance to give the best experience for priority consumers by using the system's resources efficiently. If we exploit all resources efficiently, this will provide enormous value to the development of caching techniques. Because of this issue, in this research, we propose a Weighted Cache based on Popularity and Class (Weighted-CAPIC) to improve the performance of the main priority class and the whole system in the mobile environment.

4. Weighted Cache Based on Popularity and Class (Weighted-CAPIC) Algorithm

The utility function for content without content class distinction was carried out by Dehghan et al. [7]. In this section, we develop this equation to accommodate multiclass content. Internet traffic has different requirements; therefore, we should provide different treatment for each class of traffic. We derived the utility function formula for multiclass content to obtain the relationship between weight and cache hit ratio. This was necessary to determine the proper treatment for the priority content class.

The utility function for multiclass content uses the utility relationship to the hit probability for each content class, d, which can be written as in Equation (1)

$$U_d(p_d) = \begin{cases} \omega_d \frac{p_d}{1-\gamma}^{1-\gamma}, \gamma \geq 0 \text{ and } \gamma \neq 1 \\ \omega_d . \log_{p_d} \end{cases} \quad (1)$$

where:
p_d = hit probability for content class d
$U_d(p_d)$ = utility for class d with a hit probability p_d
ω_d = weight of class d

To maximize the utility of the content store, we can write the Equation (2)

$$max_{p_d} \sum_{d=1}^{D} U_d(p_d) \quad (2)$$

Since there are D number of content classes and we used three content classes with different requirement as in Table 1, it can be written that the relationship between the portion of each class, c_d, and the total size of content storage, C, is as Equations (3) and (4).

$$c_1 + c_2 + c_3 = C \quad (3)$$

and

$$\sum_{i=1}^{c_1} r_{i1} + \sum_{i=1}^{c_2} r_{i2} + \sum_{i=1}^{c_3} r_{i3} \leq D \quad (4)$$

Table 1. Class of content.

Class	Example	Request Duration	Request Rate
1	Real-time apps	Long (years)	Low–mid
2	User-Generated Content (UGC)	Mid (weeks)	Mid
3	Web access	Mid (weeks)	High
4	email, VoNDN	Short(days)	Not cached

Hit probability of class d with cache proportion c_d and using LFU cache replacement can be defined as Equation (5).

$$p_d = \sum_{i=1}^{c_d} r_{di} \quad (5)$$

where:
r_{di} = hit probability of content i on class d;
p_d = hit probability of class d;
c_d = cache proportion of class d.

The constraint for the utility function in the content store that stores multiclass content is as in Equation (6).

$$\sum_{d=1}^{D} (p_d . N_d) = C \quad (6)$$

Due to the neutrality for making the utility linear in using p_d, we use the value $\gamma = 0$. Therefore, the maximization of utility can be written as Equations (7) and (8).

$$max_{p_d} \sum_{d=1}^{D} \omega_d \frac{p_d}{1-\gamma}^{1-\gamma} \tag{7}$$

$$max_{p_d} \sum_{d=1}^{D} \omega_d . p_d \tag{8}$$

The number of utilities will be maximized if we set a more significant weight for the class d with a greater hit probability. When applied to the system, this will cause the maximum overall hit rate in the system. Based on the formula that has been derived from the utility function of multiclass content, the total utility will be maximized if we give the most significant weight to the priority class with the highest hit probability. Weighted-CAPIC adds this concept to Dynamic-CAPIC from previous research. We add the weight parameter for determining the cache portion of each class, where ω_d is the weight for class d. The Weighted-CAPIC formula determines the cache portion for class d, as in Equation (9).

The Weighted-CAPIC formula ensures that the weight is assigned to the class that requires it according to real-time conditions. The cache portion of one class can change over time, and the Weighted-CAPIC formula assigns a certain weight to the class according to what it needs in a certain real-time period. These algorithms can be implemented in the NDN mobile node because every router node can run this algorithm individually without coordinating with other nodes.

As in Figure 1, when the interest message enters, the router will check its content store. If the content requested exists in the content store, it will be sent to the consumer by the router, and the copy will be kept in the one-hop downstream router. For every request, content gets closer to the consumer. The popularity of content determines on which nodes it will be stored in the network.

The next step is router checks how much cache portion for keep the content in the router. This portion is determined based on the content class, using the Weighted-CAPIC formula. The Weighted-CAPIC algorithm calculates the total content requested over a specific frequency limit factor for each content class. The value of this parameter is described as X_d. After that, the router will sort the values and assign a weight, ω_d, corresponding to them. The router further calculates the cache portion for the content class according to Equation (9). If the cache capacity for a class is full, the algorithm will run LRU as a cache replacement method. If the cache capacity for a class is full, the algorithm will run LRU as a cache replacement method. The router performs this two-step mechanism every time there is an incoming packet of interest and data (content).

$$c_d = \frac{X_d . \omega_d}{\sum_{y=1}^{D} X_y . \sum_{m=1}^{D} \omega_m} C \tag{9}$$

where:
X_d = the number of variations in content class k that is requested more than a certain Frequency value;
c_d = cache proportion for class d;
C = total content store capacity;
D = number of content classes.

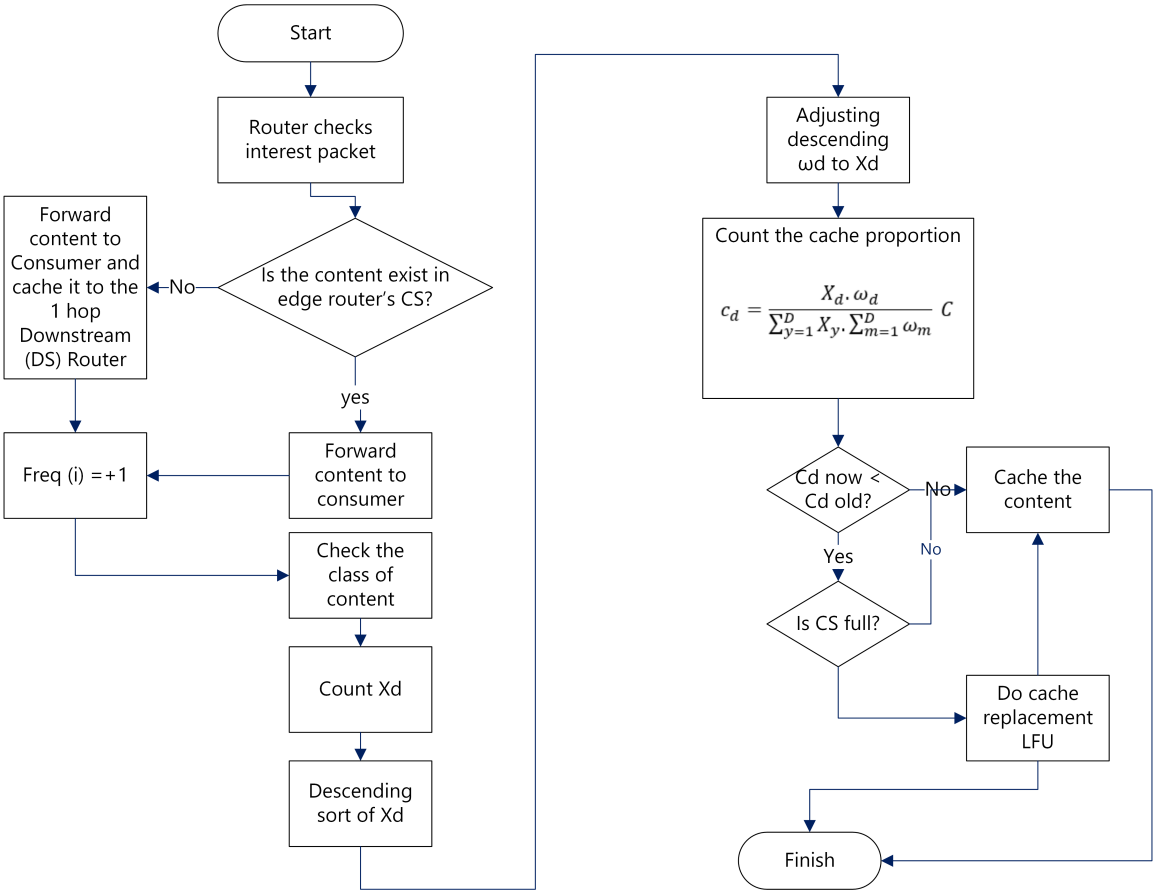

Figure 1. Flowchart of Weighted CAPIC.

5. System Model

Based on the Global Internet Phenomena Report 2019, internet traffic accessed by consumers can be divided into several groups with different characteristics [2]. Abane et al. conducted research using the Zipf–Mandelbrot distribution to model content popularity [10]. Following the RFC 7945, the Zipf–Mandelbrot distribution is suitable for the network where nodes can store content locally [17]. The average of consumer request rate is modeled using Poisson distribution.

In this research, the probability of connection between nodes is 50%. This modeling causes topology changes in a network for every simulation round. This effect the cache conditions and change dynamically. Content popularity is modeled by Zipf–Mandelbrot with exponential factor 0.8 and flattened factor 3. The average number of requests for each class follows the Poisson distribution and traffic class, according to Table 1. Simulation parameters are in detail according to Table 2.

Two experimental scenarios were carried out, with variations in the number of request levels and variations in the request rate. In the scenario of variation in number request level, classes 1, 2, and 3 each have a demand average of 10, 20, and 30, respectively. In the scenario of variations in the request rate, the number of request levels is set to 6. This means that the user request pattern changes six times during the simulation time.

In the first scenario that simulated the various number of request levels, there were three types of request levels, i.e., number of requests levels 2, 4, and 6. The number of requests level 2 means the pattern of consumer demand changed two times. For this scenario, there are two requests patterns: (10, 20, 30) and (20 30 10). This means that at half the simulation time, we used the (10, 20, 30) pattern. The (10, 20, 30) pattern implies that the number of requests for the first class is 10 requests per simulation round, the second class is 20 requests per second, and the third class is 30 requests per simulation round. As for the other half time of the simulation, we used (20, 30, 10) pattern. The number of requests

levels 4 and 6 also worked in the same way as the previous mechanism. In the scenario of 'number of requests level' 4, there were four changes in the request pattern during the simulation. Similarly, for the 'number of request level' 6, there were six changes in the pattern of requests during the simulation.

Table 2. Simulation parameters.

Parameter	Value
Number of routers	50
Content store proportion (class 1, 2, and 3)	1. Weighted-CAPIC and Dynamic-CAPIC: dynamic according to the formula 2. Static-CAPIC: 5%, 10% and 15% of total content number, normalized to total capacity of 150
Number of content	500 (1st class) 700 (2nd class) 1000 (3rd class)
Number of rounds	10.000
Frequency Limit Factor	1600
Number of request level	2 times: (10, 20, 30) and (20 30 10) 4 times: (10, 20, 30), (20 10 30), (30 20 10), and (10 30 20) 6 times (10, 20, 30), (20 10 30), (30 20 10), (20 30 10), (30 10 20) and (10 30 20)
Difference in request rate	10, 20, 30 (difference: 10) 10, 30, 50 (difference: 20) 10, 40, 70 (difference: 30) 10, 50, 90 (difference: 40)

The second scenario simulates the different request rate gaps between content classes. The term "Difference in request rate" means a difference (gap) between the request rates for classes 1, 2, and 3. In this scenario, the values are tested starting from 10, 20, 30, and 40. A 'differences in request rate' of 10 means that class 1 has a request rate of 10 requests per simulation round, then the second class increases by 10 compared to the first class, which is 20 requests per simulation round, the third class increased by 10 compared to the second class, which is 30 requests per simulation round. Likewise, the 'difference in request rate' values of 20, 30, and 40 are modeled as more significant gaps between the request rates for each content class.

System performance is seen from the cache hit ratio and path stretch parameters. The cache hit ratio parameter compares the number of requests that can be responded directly by the router node to the total number of requests. The path stretch parameter indicates the number of hops taken until a request receives data (content) as a response. Weighted-CAPIC performance is compared to Dynamic-CAPIC, Static-CAPIC, and LCD+Sharing schemes. LCD+Sharing scheme combines two typical cache schemes in NDN, namely LCD (leave copy down) and sharing scheme. We test the Weighted-CAPIC, Dynamic-CAPIC, Static-CAPIC, and LCD+sharing method in the same environment to obtain the appropriate comparison.

6. Simulation Results and Analysis

The number of request levels shows how often the pattern of consumer demand changes during the simulation time of request pattern change. Request level 6 means that the pattern of consumer demand changes six times. The higher the request level, the more often the demand pattern changes. Weighted-CAPIC provides the highest cache hit ratio for class 1, which is on average 46% larger than Dynamic-CAPIC, four times

larger than Static-CAPIC, and two times larger than LCD+Sharing, as in Figure 2. This result is in accordance with the goal of the Weighted-CAPIC algorithm, which provides the highest performance for the class with the highest priority. The result also proves that Weighted-CAPIC can accommodate dynamically changing traffic patterns in the system. Class 1 is a priority class. It has a longest request duration as in Table 1. The information has a long time to go before it becomes obsolete. The content is usually still requested over the years. Therefore, storing content in the network will provide an advantage for these repeated content requests.

The Weighted-CAPIC path stretch has the same value as Dynamic-CAPIC, and its value is the smallest compared to Static-CAPIC and LCD+Sharing schemes. With the same path stretch, Weighted-CAPIC provides the larger cache hit ratio compared to Dynamic-CAPIC.

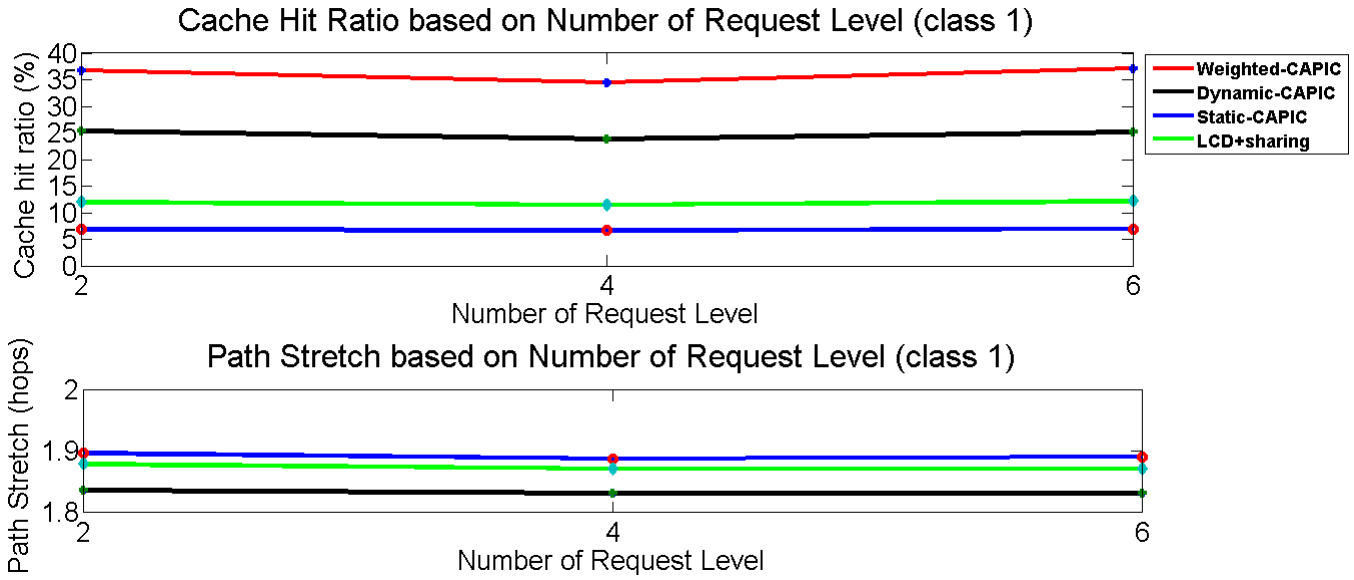

Figure 2. Cache hit ratio and path stretch of class 1 for various level of request.

The second class cache hit ratio for Weighted-CAPIC is 14.6% smaller than LCD+Sharing, 7% smaller than Dynamic-CAPIC, and 2% larger than Static-CAPIC, as in Figure 3. The second class with UGC traffic type is the second priority, after real-time traffic. So, based on the formula, Weighted-CAPIC provides a smaller portion of storage for this second class compared to the first class. This can also be regarded as compensation, because Weighted-CAPIC has given a more significant cache portion for the first class, which is a priority class. The content store capacity is fixed in every NDN router, and the cache portion must be adjusted based on the content class's needs. In Weighted-CAPIC, this adjustment is made using a formula that is ran on each NDN router.

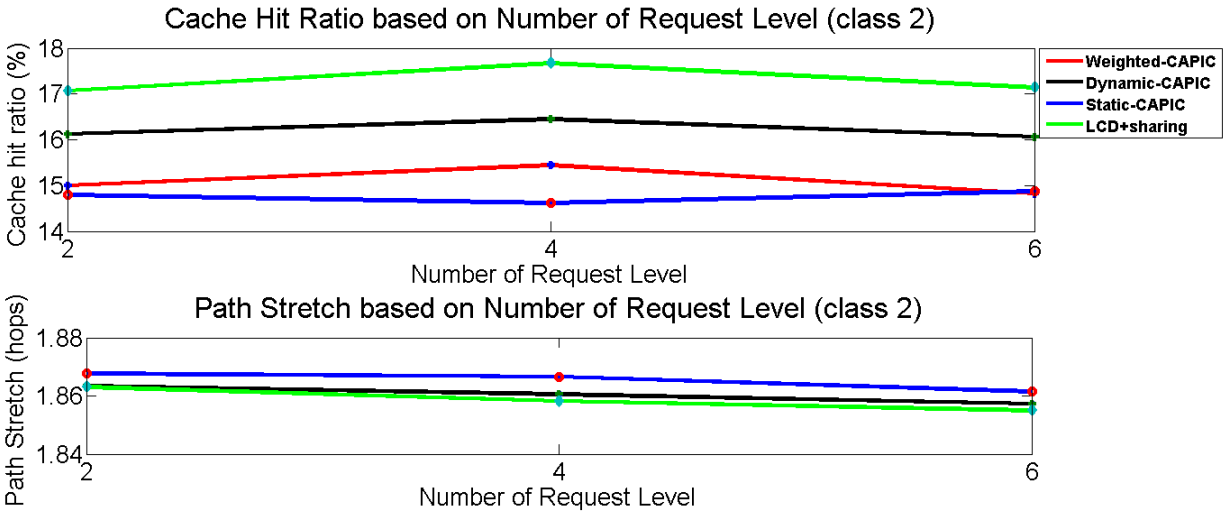

Figure 3. Cache hit ratio and path stretch of class 2 for various level of request.

The Weighted-CAPIC path stretch has the same value as Dynamic-CAPIC, having a smaller value of about 0.2% than Static-CAPIC. In this second class, the Weighted-CAPIC path-stretch value is higher than the first class, which means that it is necessary to pass longer distances to obtain the desired content for this second class.

Weighted-CAPIC gives the smallest cache hit ratio value for class 3, as in Figure 4, because the priority of the third class is the lowest. In addition, by looking at Figures 2–4, it can be seen that Weighted-CAPIC provides the lowest cache hit ratio for the third class compared to first and second. Again, this corresponds to the priorities of classes 1, 2, and 3. Path stretch decreases when the request level increases for all caching schemes.

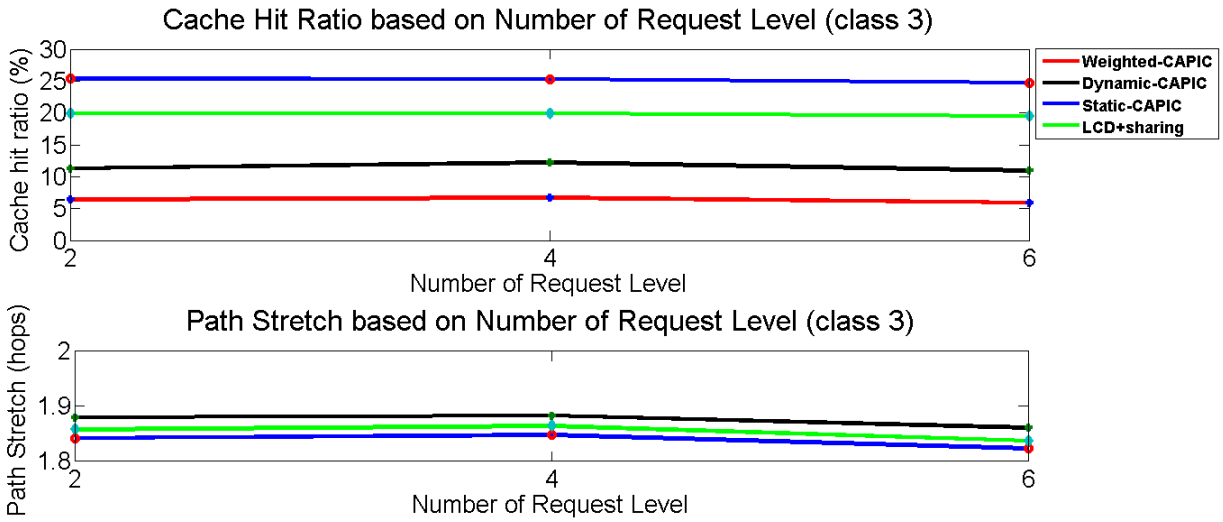

Figure 4. Cache hit ratio and path stretch of class 3 for various level of request.

Weighted-CAPIC provides the highest network cache hit ratio parameter. This parameter involves the overall cache hit ratio for all traffic classes in the network/system. Figure 5 shows that the higher the number of request levels, the higher the cache hit ratio of the Weighted-CAPIC and Dynamic-CAPIC schemes. Weighted-CAPIC cache hit ratio is higher than Dynamic-CAPIC, and at request level 6, Weighted-CAPIC gives the most significant cache hit ratio, which is 11% higher than Dynamic-CAPIC. Static-CAPIC provides the smallest cache hit ratio for the higher request levels. This condition happens because Static-CAPIC cannot adjust its rules when requests come in arbitrary patterns.

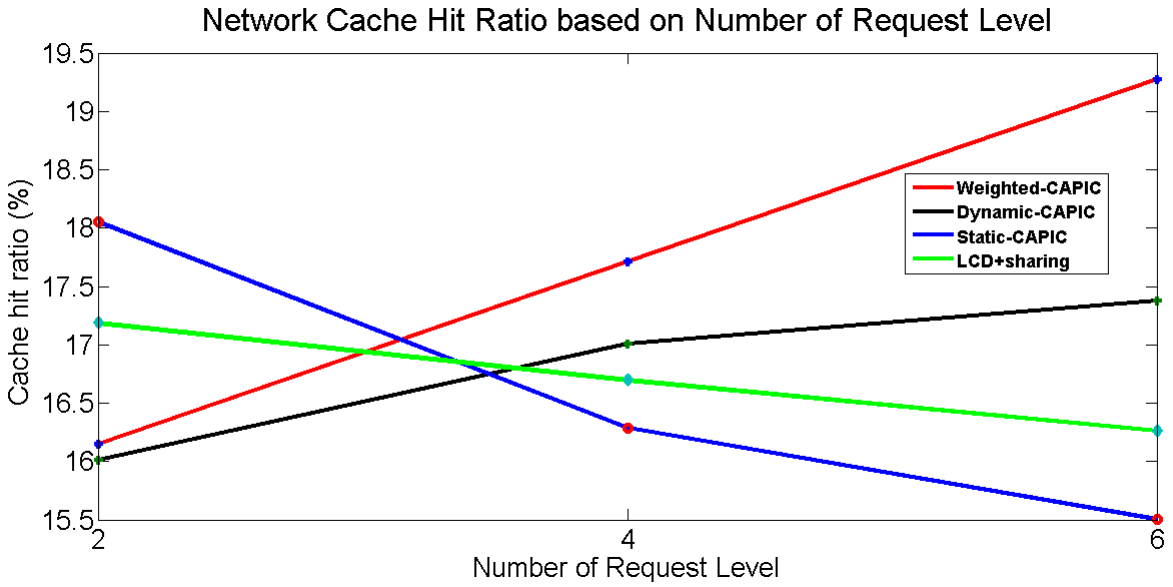

Figure 5. Network cache hit ratio for various level of request.

Algorithm 1: FunctionWeighted-CAPIC

Initialization;
D: number of content class ;
i: content ;
C: capacity of content store ;
Function WeightedCAPIC(input: D, C, i) ;
if *request=i* **then**
  Check last-location i;
  if hop_i*!=1* **then**
    hop_i=hop_i-1 ;
  else
    hop_i=1
else
  end process
% calculate number of content variation for each class;
for $m = 1 : D$ **do**
  $Total_X$ =+ X_m;
Sorting(Xm) **for** $d = 1 : D$ **do**
  give the ω_d based on X_m
% calculate cache portion of content class d;
$c_d = \frac{X_d.\omega_d}{\sum_{i=1}^{D} X_i.\sum_{m=1}^{d} \omega_m} C$;
Check capacity of c_d;
if *capacity not full* **then**
  save i in part-of-Content-store;
else
  do replacement algorithm;
  save i in part-of-Content-store;
send i to consumer ;

In the 'difference in request rate' scenario, Weighted-CAPIC provides the most significant value compared to other schemes for class 1. As in Figure 6, its cache hit ratio is 45% higher than Dynamic-CAPIC. The cache hit ratio of Weighted-CAPIC is fluctuates 1–4% around its average value and decreases further for more considerable request rate

differences. This means that Weighted-CAPIC can accommodate the various consumer request rates, with the low or high gap between each traffic class. Dynamic-CAPIC gives the cache hit ratio variation 1–3% around the average value and falls for the more significant request rate difference value. It drops dramatically compared to the average, with the difference in request rate increasing up to 50%. At the same time, Static-CAPIC is more stable with a difference of only 1% compared to the average value.

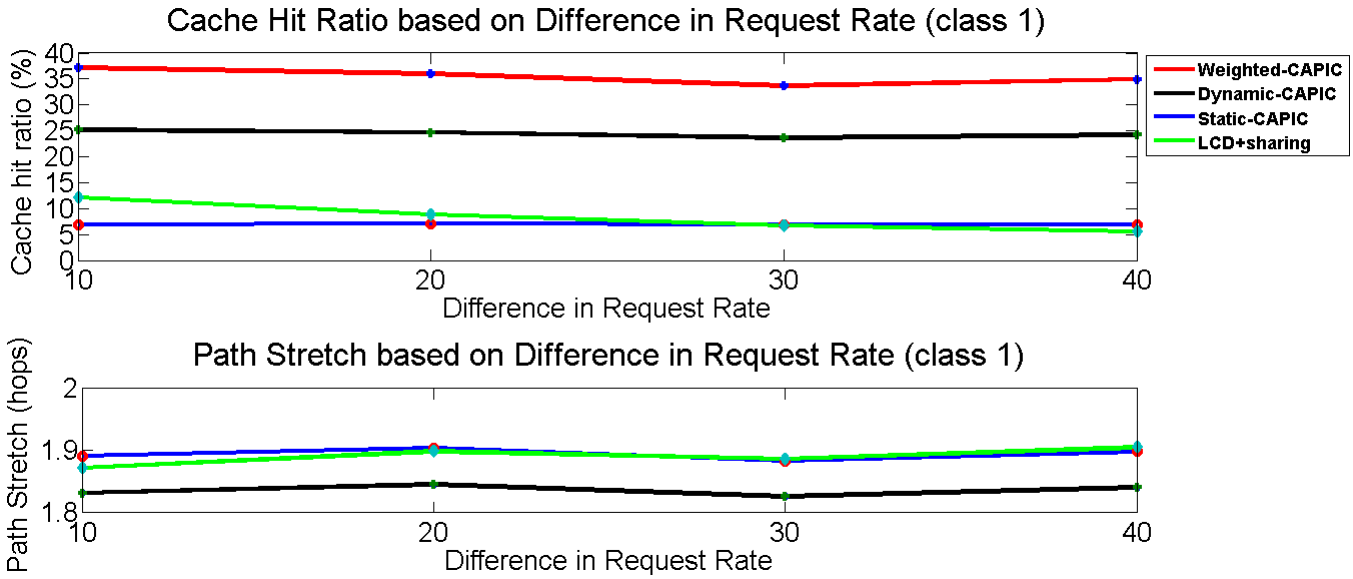

Figure 6. Cache hit ratio and path stretch of class 1 for various request rate difference.

For the path stretch parameter, Weighted-CAPIC has the same value as Dynamic-CAPIC. Weighted-CAPIC provides the lowest path stretch for this first class. It means that content can be obtained with the lowest hop distance, shorter than the other strategies. Weighted-CAPIC gives the smallest path stretch compared to the all comparison scheme.

LCD+Sharing provides the largest cache hit ratio for the second class, followed by Dynamic-CAPIC and Weighted-CAPIC. This happens because the second class has a lower priority than the first class, so Weighted-CAPIC provides a smaller cache portion and results in a smaller cache hit ratio. The Weighted-CAPIC path stretch is the same as Dynamic-CAPIC and is between the Static-CAPIC and LCD+Sharing stretch path values, as shown in Figure 7. In the second class, Weighted-CAPIC provides a relatively more extensive path stretch for increasing the demand gap between classes. This means that the greater the request rate, the higher the number of hops that must be passed to obtain the desired data. This happens because the greater the number of requests, the greater the variety of data requested.

The largest cache hit ratio is given by the Static-CAPIC scheme for class 3, as shown in Figure 8, because it has the highest rate compared to other classes. Static-CAPIC has accommodated this condition by providing the largest cache portion in class 3. The cache hit ratio of Weighted-CAPIC fluctuates around 3–8% of the average value. The cache hit ratio of Dynamic-CAPIC fluctuates about 1–4% of the average value. The LCD+Sharing cache hit ratio increases along with the rise in request rate difference, up to 12% of the average for the largest request gaps. Weighted-CAPIC provides the lowest cache hit ratio compared to other comparison schemes for the third class. This happens because, in Weighted-CAPIC, the third class is the lowest priority class; therefore, it is given the minor cache portion compared to the first and second class. This is different from the Static-CAPIC and LCD+Sharing schemes, which provide greater performance for this third class. Weighted-CAPIC provides a lower cache hit ratio than Dynamic-CAPIC in this third class as compensation for delivering greater performance to the classes with higher priority. The

Weighted-CAPIC path stretch is the same as Dynamic-CAPIC, which is the largest stretch path, followed by LCD+sharing and Static-CAPIC scheme.

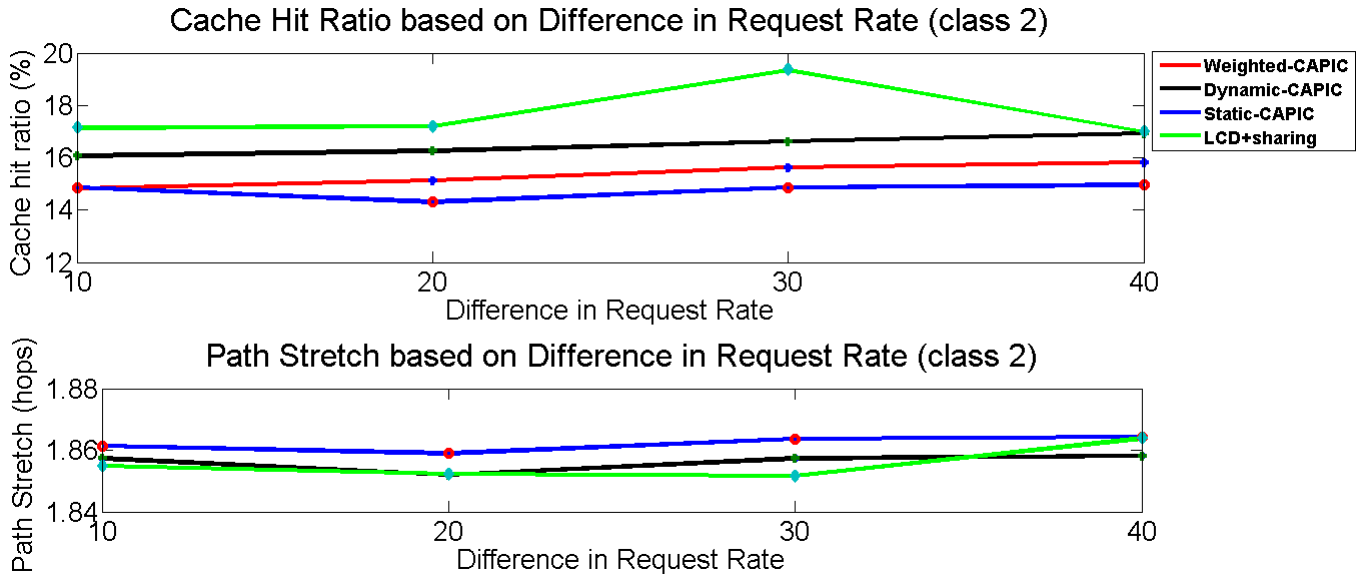

Figure 7. Cache hit ratio and path stretch of class 2 for various request rate difference.

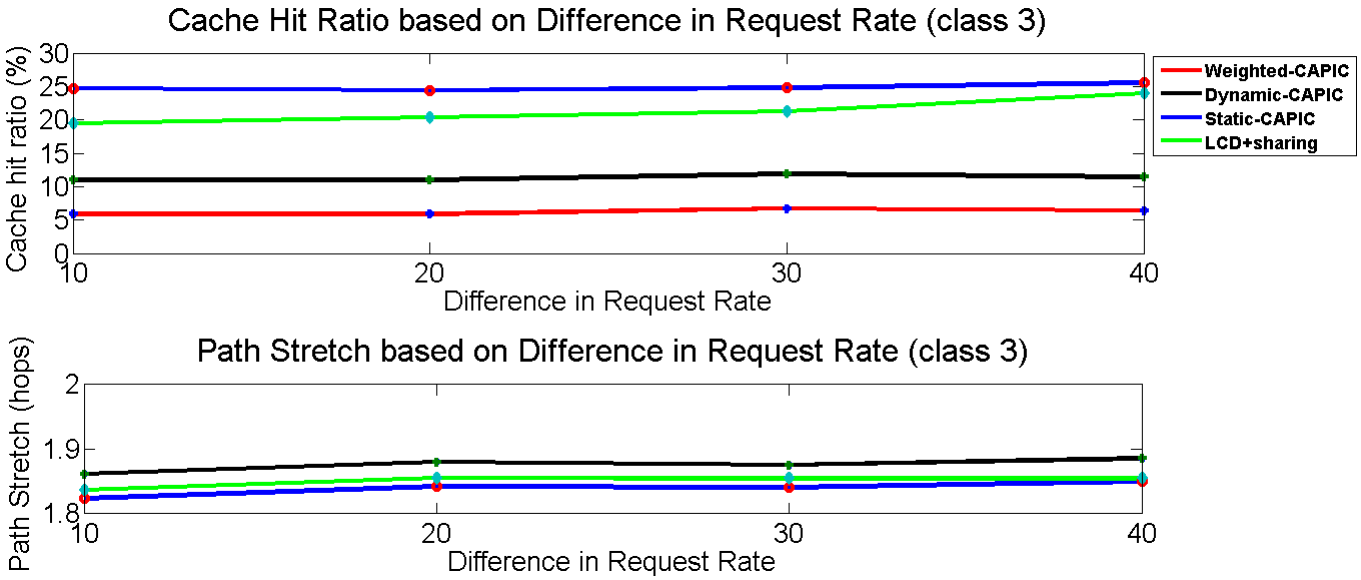

Figure 8. Cache hit ratio and path stretch of class 3 for various request rate difference.

The simulation scenario with variations in the request rate difference gives the largest network cache hit ratio value for Weighted-CAPIC, followed by Dynamic-CAPIC, LCD+Sharing, and Static-CAPIC. As in Figure 9, Weighted-CAPIC provides an average network cache hit ratio of 9.7% higher than Dynamic-CAPIC, 22% higher than LCD+Sharing, and 24% higher than Static-CAPIC scheme. The network cache hit ratio is the total cache hit ratio for the system. Figure 9 shows that Weighted-CAPIC not only provides the higher performance to the priority class but also increases the total system cache hit ratio, compared to the Dynamic-CAPIC, Static-CAPIC, and LCD+sharing schemes.

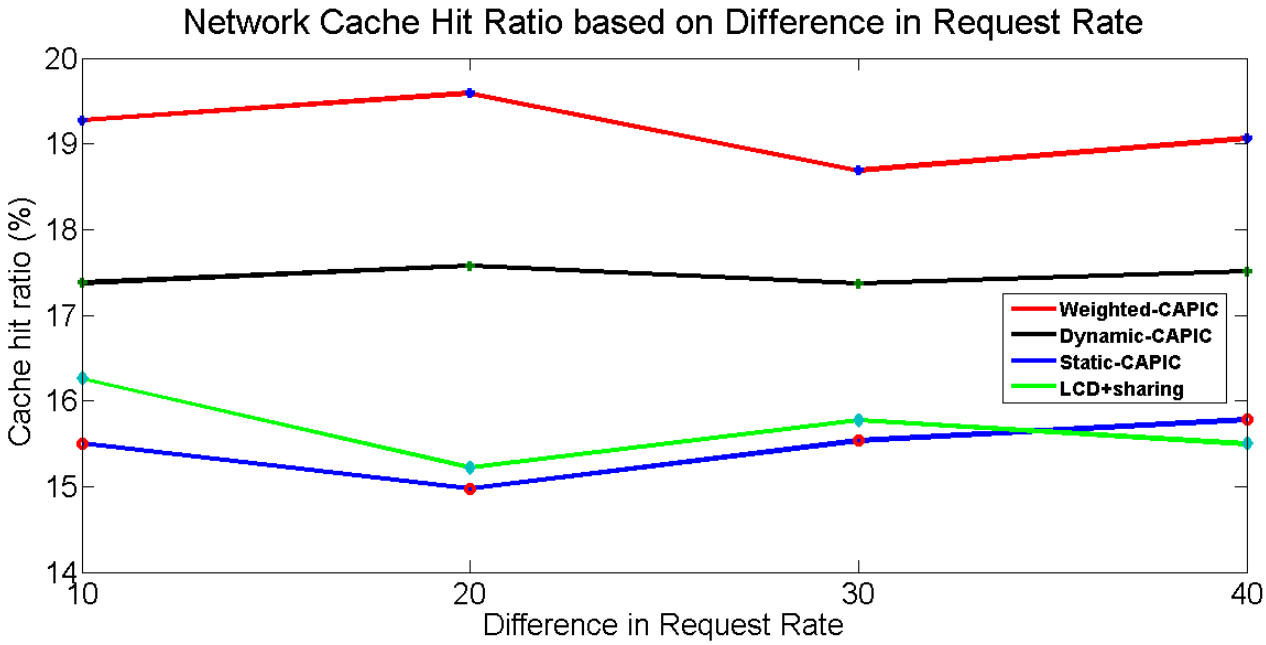

Figure 9. Network cache hit ratio for various request rate difference.

From the simulation results, Weighted-CAPIC gives the higher cache hit ratio to the priority class. Weighted-CAPIC improved the method in the Dynamic-CAPIC algorithm. Dynamic-CAPIC can dynamically allocate cache portions for each traffic class based on consumer demand. This is done using the Dynamic-CAPIC formula, which provides the cache portion based on the variations in content for certain classes. However, based on our derived multiclass-content utility function, we must treat priority classes differently besides just considering the traffic variation. The Weighted-CAPIC formula is also as dynamic as the Dynamic-CAPIC formula, which can provide different portions at any time, according to consumer demand patterns. However, Dynamic-CAPIC has not provided the most extensive resource for the highest priority class. The highest priority is given to the class which has the highest opportunity to be requested by consumers at one time.

For this reason, Dynamic-CAPIC was improved into Weighted-CAPIC, which focused more on providing larger resources to the appropriate class. The simulation results for Weighted-CAPIC show that this algorithm successfully provides a higher cache hit ratio for the priority class and provides a higher cache hit ratio for the network system. This performance is supported while the algorithm still provides the same path stretch value as Dynamic-CAPIC.

7. Algorithm Complexity

To analyze the Weighted-CAPIC performance, we conducted several tests and also explored it in terms of algorithm complexity. We used the Time Complexity parameter to measure the complexity of the algorithm. Measurement of complexity based on time is carried out for a particular input [18]. However, the processing time is highly dependent on various factors such as the type of machine used, the parallel processing performed, and so on. Big-O notation is used to represent it to avoid biased algorithm complexity values.

Weighted-CAPIC, according to the pseudocode in the algorithm 1, can be written as the input correlation of total class content D, equal to 5D+2, so that the Big-O notation is O(D). From the previous research, Dynamic-CAPIC caching algorithm as input correlation can be written mathematically as 4D+2, and its Big-O notation is O(D) [9]. This means that the complexity of the Weighted-CAPIC algorithm is the same as Dynamic-CAPIC but provides a better cache hit ratio (CHR) than Dynamic-CAPIC for the priority class and the overall network.

8. Conclusions

This research proposed the Weighted-CAPIC caching algorithm to improve the priority class performance. Weighted-CAPIC works by adjusting different weights for every content class besides considering the variation of the content in providing a cache portion in the NDN router. Weighted-CAPIC provides the highest cache hit ratio for the priority class and the network, outperforming the Dynamic-CAPIC, Static-CAPIC, and LCD+Sharing scheme. The performance evaluation shows that the greater the number of request levels (the more dynamic consumer demand), the greater the total cache hit ratio provided by Weighted-CAPIC. Weighted-CAPIC can accommodate dynamic consumer demands more than all comparison algorithms in this study, i.e., Dynamic-CAPIC, Static-CAPIC, and LCD+Sharing. It is also very good at accommodating the various consumer request rates, with the low or high gap between each traffic class, and provides better performance with the same complexity as Dynamic-CAPIC. Weighted-CAPIC is also suitable for systems with dynamic consumer demand patterns and has a high priority gap for each traffic class. In this condition, Weighted-CAPIC can provide higher performance on certain priority classes. Weighted-CAPIC also provides a higher network/system cache hit ratio than Dynamic-CAPIC and all comparison algorithms, which means that Weighted-CAPIC allocates the cache portion for every class efficiently. But as compensation, Weighted-CAPIC reduces the cache portion of other classes that are of a lower priority.

9. Future Research

We will combine the Weighted-CAPIC with machine learning to determine the weight of each content class as the future research.

Author Contributions: Conceptualization, L.V.Y. and N.R.S.; methodology, L.V.Y. and N.R.S.; software, L.V.Y. and I.J.M.E.; formal analysis, L.V.Y. and N.R.S.; writing original draft, L.V.Y.; validation, N.R.S. and I.J.M.E.; writing—review editing, N.R.S.; supervision, N.R.S. and I.J.M.E.; resources, L.V.Y. and I.J.M.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable, the study does not report any data.

Acknowledgments: This research was supported by Telkom University.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

NDN	Named Data Network
UGC	User-Generated Content
CHR	Cache Hit Ratio
CS	Content Store
PIT	Pending Interest Table
FIB	Forwarding Information Based
FIFO	First In, First Out
LRU	Least Recently Used
LFU	Least Frequently Used
LCD	Leave Copy Down

References

1. Cisco. *Cisco Annual Internet Report (2018–2023)*. Technical Report. 2020. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed on 19 February 2022).
2. Sandvine. *The Global Internet Phenomena Report*; Technical Report. September 2019. Available online: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/Internet%20Phenomena/Internet%20Phenomena%20Report%20Q32019%2020190910.pdf (accessed on 19 February 2022).

3. Shannigrahi, S.; Fan, C.; Partridge, C. What's in a Name?: Naming Big Science Data in Named Data Networking. In Proceedings of the ICN 2020—Proceedings of the 7th ACM Conference on Information-Centric Networking, Virtual Event, Canada, 29 September–1 October 2020; pp. 12–23. [CrossRef]
4. Jin, H.; Xu, D.; Zhao, C.; Liang, D. Information-centric mobile caching network frameworks and caching optimization: A survey. *Eurasip J. Wirel. Commun. Netw.* **2017**, *33*, 1–32. [CrossRef]
5. Kim, Y.; Kim, Y.; Bi, J.; Yeom, I. Differentiated forwarding and caching in named-data networking. *J. Netw. Comput. Appl.* **2016**, *60*, 155–169. [CrossRef]
6. Sourlas, V. Partition-based Caching in Information-Centric Networks. In Proceedings of the Seventh IEEE International Workshop on Network Science for Communication Networks (NetSciCom 2015), Hong Kong, China, 27 April 2015; pp. 396–401.
7. Dehghan, M.; Massoulie, L.; Towsley, D.; Menasche, D.S.; Tay, Y.C. A Utility Optimization Approach to Network Cache Design. *IEEE/ACM Trans. Netw.* **2016**, *27*, 1013–1027. [CrossRef]
8. Majd, N.E.; Misra, S.; Tourani, R. Split-Cache: A holistic caching framework for improved network performance in wireless ad hoc networks. In Proceedings of the 2014 IEEE Global Communications Conference, GLOBECOM 2014, Austin, TX, USA, 8–12 December 2014; pp. 137–142. [CrossRef]
9. Yovita, L.V.; Syambas, N.R.; Joseph, I.; Edward, M.; Kamiyama, N. Performance Analysis of Cache Based on Popularity and Class in Named Data Network. *Future Internet* **2020**, *12*, 227. [CrossRef]
10. Abane, A.; Daoui, M.; Bouzefrane, S.; Banerjee, S.; Abane, A.; Daoui, M.; Bouzefrane, S.; Banerjee, S.; Realistic, P.M.A.; Abane, A.; et al. A Realistic Deployment of Named Data Networking in the Internet of Things to cite this version: HAL Id: Hal-02920555 a Realistic Deployment of Named Data Networking in the Internet of Things. *J. Cyber Secur. Mobil.* **2020**, *9*, 1–27.
11. Afanasyev, A.; Burke, J.; Wang, L.; Zhang, B. A Brief Introduction to Named Data Networking. In Proceedings of the MILCOM 2018—2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; IEEE: Piscataway, NJ, USA, 2018.
12. Conti, M.; Member, S.; Gangwal, A.; Hassan, M.; Lal, C.; Losiouk, E. The Road Ahead for Networking: A Survey on ICN-IP Coexistence Solutions. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2104–2129. [CrossRef]
13. Shailendra, S.; Sengottuvelan, S.; Rath, H.K.; Panigrahi, B.; Simha, A. Performance evaluation of caching policies in NDN-an ICN architecture. In Proceedings of the IEEE Region 10 Annual International Conference, Proceedings/TENCON, Penang, Malaysia, 5–8 November 2017; pp. 1117–1121. [CrossRef]
14. Jing, Y. *Evaluating Caching Mechanisms in Future Internet Architectures*; Technical report; Massachusetts Institute of Technology: Cambridge, CA, USA, 2016.
15. Chu, W.; Dehghan, M.; Towsley, D.; Zhang, Z.L. On allocating cache resources to content providers. In Proceedings of the ACM-ICN 2016—Proceedings of the 2016 3rd ACM Conference on Information-Centric Networking, Kyoto, Japan, 26–28 September 2016; pp. 154–159. [CrossRef]
16. Yovita, L.V.; Syambas, N.R.; Matheus Edward, I.Y. CAPIC: Cache based on Popularity and Class in Named Data Network. In Proceedings of the 2018 International Conference on Control, Electronics, Renewable Energy and Communications, ICCEREC 2018, Bandung, Indonesia, 5–7 December 2018; pp. 24–29. [CrossRef]
17. Pentikousis, K.; Bari, P. *RFC7945: Information-Centric Networking: Evaluation and Security Considerations*. Technical Report. 2016. Available online: https://www.hjp.at/doc/rfc/rfc7945.html (accessed on 19 February 2022).
18. Mala, F.A.; Ali, R. The Big-O of Mathematics and Computer Science. *J. Appl. Math. Comput.* **2022**, *6*, 1–3. [CrossRef]