future internet

MDPI

Article

Performance Analysis of Cache Based on Popularity and Class in Named Data Network

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

[1] School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Kota Bandung 40132, Indonesia; nana@stei.itb.ac.id (N.R.S.); ian@stei.itb.ac.id (I.J.M.E.)
[2] Department of Electronics Engineering and Computer Science, Fukuoka University, Fukuoka 814-0180, Japan; kamiyama@fukuoka-u.ac.jp
* Correspondence: leanna@students.itb.ac.id; Tel.: +62-22-250-2260

Received: 13 November 2020; Accepted: 7 December 2020; Published: 9 December 2020

Abstract: The communication network is growing with some unique characteristics, such as consumers repeatedly request the same content to the server, similarity in local demand trend, and dynamic changes to requests within a specific period. Therefore, a different network paradigm is needed to replace the IP network, namely Named Data Network (NDN). The content store, which acts as a crucial component in the NDN nodes is a limited resource. In addition, a cache mechanism is needed to optimize the router's content store by exploiting the different content services characters in the network. This paper proposes a new caching algorithm called Cache Based on Popularity and Class (CAPIC) with dynamic mechanism, and the detail explanation about the static method also presented. The goal of Static-CAPIC was to enhance the total cache hit ratio on the network by pre-determining the cache proportion for each content class. However, this technique is not appropriate to control the cache hit ratio for priority class. Therefore, the Dynamic-CAPIC is used to provide flexibility to change the cache proportion based on the frequency of requests in real-time. The formula involves considering the consumers' request all the time. It gives a higher cache hit ratio for the priority content class. This method outperforms Static-CAPIC, and the LCD+sharing scheme in the total network cache hit ratio parameter and channels it to the priority class.

Keywords: Named Data Network; caching; popularity; content class; priority

1. Introduction

The increase in a bit rate of multimedia traffic makes the TCP/IP architecture inefficient in delivering time-sensitive multimedia traffic. We need the content-based network instead of a host-based network [1]. Named Data Network (NDN) is a content-centric network that utilizes network resources to store data on routers [2]. The whole network resources need to support communication maximally. NDN allows consumers to find the data requested at the nearest router. This mechanism reduces the overall network load and potential delay. Furthermore, the placement of content at a certain node that is closer to the consumer causes communication to be more efficient, while router nodes independently place content in its content store without coordinating with others on the network. This mechanism also makes NDN strongly support consumer mobility with content requests based on the name, and not on a specific host address [2,3]. The network responds to a consumer request in the aggregate, and it makes communication more efficient.

Nowadays, communication occurs with some unique characteristics [4], such as consumers repeatedly asking for the same content from the server, the character is similar to the local environment, and dynamic changes in demand over time. The IP network has become inefficient because the

consumer is only served by the server, despite the distance. This system is called host-based communication. In the IP network, the consumers request content to a specific address. Therefore, a shift to the host-based paradigm into the content-based, namely Named Data Network (NDN), is needed because what is needed by the consumer is content, not the server. NDN makes the process of sending data more efficient than IP networks because, it allows consumers to transfer interest packets to the system, and not to a specific IP address [5,6]. NDN routers can store data/content in its content store, and when an interest packet containing the consumer's request enters an NDN router, it checks whether the requested content exists on its content store. When the router has the requested data, it immediately sends a copy to the consumer, and when it does not, it checks the Pending Interest Table (PIT). When the PIT has no information on the requested data, the router adds this request information and the information of the sending side. The request is therefore forwarded to the next router using the information in the Forwarding Information Base (FIB) until it identifies the desired content.

One of the essential components of the NDN router with a limited resource is the content store. Therefore, a caching mechanism capable of optimizing the content stores is needed by considering the difference in traffic character on the network. The size of the content store [7] and the cache policy [8] affect the system performance. Several studies have been conducted on energy consumption [9,10].

Yovita et al. explained that research on the caching mechanism in NDN that distinguishes content treatment for different services is vital [11]. Kim et al. researched the differentiation of service classes on NDN. These studies discussed differentiating service classes on IP, such as DiffServ, which is adjusted and used in NDN [12]. In this scheme, content groups have been distinguished with differing interest packet's marking rate depending on the class, without special treatment on the cache. Furthermore, differentiation of package treatment is carried out on its forwarding activities. Wu et al. [13] stated that the difference in the treatment of data is carried out by statically and manually distinguishing its lifetime for each class.

Zhang et al. carried out a research that formulated caching placement into optimizing problem in accordance with the imbalance of caching gain caused by different content popularity [14]. Zhang et al. further proposed a scheme regarding interdomain traffic saving and weighted hop saving. Carofiglio et al. [15] conducted research on the content store settings for different applications. The data is saved based on Time to Live (TTL) for each application type, with several methods analyzed by setting a static memory allocation, determining the priority application, and calculating the amount of allocated memory based on the application's weight. In addition, the other class can utilize a proportion, when the capacity is not full.

The other research related to the traffic characteristics were explained by Cho et al. [16,17]. They stated that there are variations in requirements and traffic patterns on the network. The 2020 Mobile Internet Phenomena Report shows that approximately 65% of the worldwide mobile downstream traffic is video related-content, and 12.7% is social networking traffic [18]. Therefore, the process of selecting traffic that needs to be cached is still an open issue. The development of caching algorithms until 2019 focused on popularity and probabilistic-based caching [19].

Presently, there are no caching techniques that pay attention to the popularity and exploit the characteristics of content class to maximize network cache hit ratio. NDN routers require the caching method to autonomously and independently determine where the content is to be cached and also the portion. The role is potent in influencing cache hits and hop lengths, which affect network delay and load. Therefore a technique capable of differentiating and optimizing the treatment based on the content classes is needed. This technique is expected to consider the popularity of the content in determining the cache proportion at the router's content store, in order to provide better performance in NDN.

This research aims to develop a new caching algorithm, named Cache Based on Popularity and Class (CAPIC), that is capable of providing answers to the problems previously described and improve the overall NDN network performance. The detailed mechanism of the CAPIC algorithm is explained in the next section. Two different mechanisms were applied to the CAPIC algorithm, namely Static and Dynamic. The purpose of Static-CAPIC is to enhance the total network cache hit ratio using the simple technique [20]. We explain Static-CAPIC mathematically and analyze its performance. Then we define the improvement to Dynamic-CAPIC to accommodate the demand more flexibly. The Static-CAPIC and Dynamic-CAPIC are similar at the global mechanism, but they are different in cache proportion determination. Static-CAPIC has the static proportion for cache, while Dynamic-CAPIC specifies the cache proportions dynamically and in accordance with the Dynamic-CAPIC proportion formula. The proportion can change every time according to the actual consumer request rate.

This research is written in the following order. Section 2 discusses the general scheme of Cache Based on Popularity and Class (CAPIC). Section 3 explains the Static-CAPIC, including mathematical analysis for the partitioned content store. Section 4 describes Dynamic-CAPIC, a modification of Static-CAPIC with a dynamic cache proportion computation mechanism for each content service class. Section 5 simulates the model and test results, while Section 6 concludes the research. Section 7 is future research.

2. Basic of Cache Based on Popularity and Class

There are two types of packages in NDN, namely interest packet and data packet. The interest packet contains a request for content (data) and consumers send it to the network. The data packet includes contents that are requested by consumers. The interest packet is sent by the consumer, while the producer or the router sends the data packet. The terms 'interest packet' and 'data packet' are used often in this paper. The name 'data' means 'content' and can be interchangeable. In NDN, the router or producer node sends the data packet to consumers only when consumers first send the interest packet to inform what content they need. This process is efficient and fast when the data location is close to them because it only involves less node or router in the network. There are lots of content in the network, therefore, the caching mechanism is needed to decide where the node is to cache the copy of data and in the amount saved on the router's content store. It is important to keep the data close as much to the consumer and provide the optimal network performance.

2.1. Background and Related Research

Caching strategies can be grouped into cache placement, content selection, and policy [4]. Cache placement is a mechanism used to decide the cache location of the data. Cache content selection determines the content to be cached or deleted from the content store. Cache policy regulates the cooperation mechanism between router nodes, coordination between nodes, etc.

One of the cache placement techniques is Leave Copy Down (LCD). This technique give an additional bit, called 'flag', on the data. When an interest packet enters the router, it changes the flag bit in the packet to '1'. The downstream router that receives the data stores a copy in the content store [21]. LCD technique places data closer to consumers for any popular content. Every time there is a consumer request, the data are being copied to the one-hop router closer to the consumer.

A caching scheme similar to LCD is used on WAVE algorithm [16], which is a data file that is divided into several smaller packages called chunks. Interest packages come from users and contain requests for content. When a consumer sends an interest packet, the content chunk 'A' is sent from the producer to the one-hop downstream router to be stored, before it is sent to the consumer. Furthermore, when the next consumer sends an interest packet to request the same content, the next downstream router saves the chunk 'A' before sending it to the consumer. Every new request for the same content shifts the chunk 'A' that needs to be kept at the router one hop closer to the consumer.

The idea of the LCD and WAVE mechanism is modified and applied to our new cache algorithm, which is Cache Based on the Popularity and Class (CAPIC) algorithm as the cache placement method. In the CAPIC algorithm, data are cached one hop closer to the consumer for each interest packet sent by the consumer. On the router, the cache policy is enforced to run on differentiated cache proportion of each content class. When the content store is full, the cache replacement technique is used to select the content to be removed. This mechanism draws the content closer to the consumer every time it is requested.

2.2. Caching Strategy of CAPIC

There are several routers involved in communication network, with its members optimally utilized to improve network performance. To accommodate this strategy, a Cache Based on Popularity and Class (CAPIC) scheme for Named Data Network (NDN) is proposed. This scheme is accommodated in two things, namely the popularity of the content and the class of content services. This means, there are two steps in the CAPIC mechanism. The first step aims to guarantee the content placement based on priority, while the second step guarantees the consideration of content class in this system. The popularity of the content determines the router the content will be cached in, while the content class determines the proportion of its cache storage. The detailed mechanism of the CAPIC algorithm is carried out in the following two steps:

1. In the first step, the content popularity determines the cache location (cache placement). Every time consumers request content, these data are sent, and a copy is stored in the one-hop downstream router that is closer to the consumer. The more popular content, the closer it is cached to the consumer. The mechanism implemented in CAPIC continues to the second step.
2. In the second step, the proportion of cache storage (cache policy) is determined on the router based on its content class. However, only a certain proportion of the router's content store is used for store certain content class data. When the section's capacity is full, data are deleted using the cache replacement technique. For the Static-CAPIC, the content store's size of each class is pre-determined, based on request pattern observation over a certain period. The changing proportion of Static-CAPIC is still carried out manually. Meanwhile, in the Dynamic-CAPIC, the cache proportion is determined dynamically following the new proposed formula of Dynamic-CAPIC, and it can change in real-time based on the consumer request pattern.

The CAPIC mechanism flowchart is shown in Figure 1. After the NDN routing protocol is established, consumers send the interest packet to request a data to the network. When the interest packet enters the NDN router, its content is checked to determine possible initial existence. Assuming the router already has the data requested by the consumer, a copy is sent to the closest router with a copy sent to the consumer. This procedure is the first step of the CAPIC algorithm that is conducted based on the content popularity and its ability to move the data one hop closer for every request until it reaches the edge router. The second step is based on the class of content. Therefore, when the router decides to save the copy of data, it needs first to calculate the amount of data a specific content class can save in its content store. This is because content that belongs to a particular class is not being stored in the storage that is intended for other classes. This second step is conducted with dependency on the type of CAPIC, that is Static or Dynamic. For the Static-CAPIC and Dynamic-CAPIC, the proportions are followed by the pre-determined size and Dynamic-CAPIC formula, respectively. This step is carried out every time the router needs to save the data. When the content store is full, the content replacement scheme is performed. In this research, the LFU mechanism is used to carry out this activity [21] because of its strength and effectiveness with request pattern following the Zipf–Mandelbrot distribution [3].

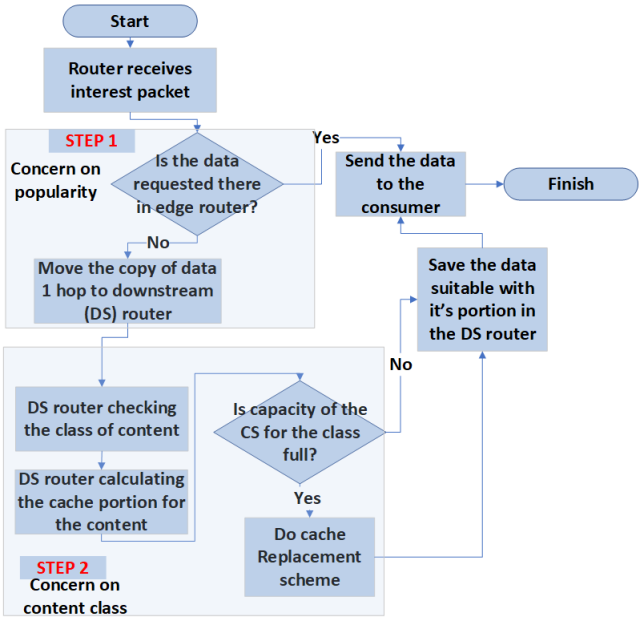

Figure 1. Flowchart of Cache Based on Popularity and Class (CAPIC) caching mechanism.

2.3. System Model

In accordance with Figure 2, the network model is made with N router members. The set member is routers r_n, where $n = 1, 2, 3, ..., N$. Router r_1, also known as an edge router, is the closest router to the consumer, while router r_N is the closest router to the producer. Parameter c_{kc} is specified as the cache size for class c, on the router r_k.

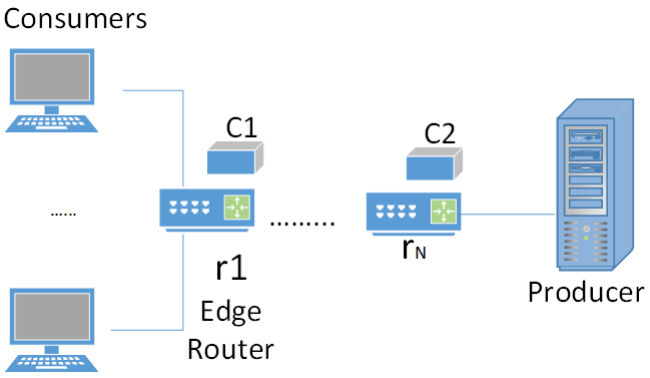

Figure 2. Network model.

The consumer demand is modeled according to the results of several previous studies. Fricker et al. provide measurement results of consumer's request to the web traffic services, User Generated Content (UGC), and video content [22]. All requests can be modeled following the Zipf-like distribution with a similar exponent factor value, which is approximately 0.8. RFC7945 used to determine the variation of the Zipf–Mandelbrots distribution. Zipf–Mandelbrot provides a better model for the environment where the node can store the requested content locally [23]. Studies on the characteristics of traffic also explain that the web content population is the largest, followed by User Generated Content (UGC) and video on demand (VoD) content [22]. One other study of the demand model was carried out by Che et al. [24] on modeling a document request based on the Poisson distribution with an average value of the request at a certain time. The demand model is as shown in Figure 3, where the average number of requests for a content service class follows the Poisson distribution with the request rate λ_k for class k, which is follows the Zipf–Mandelbrot distribution with exponent factor α_k for class k.

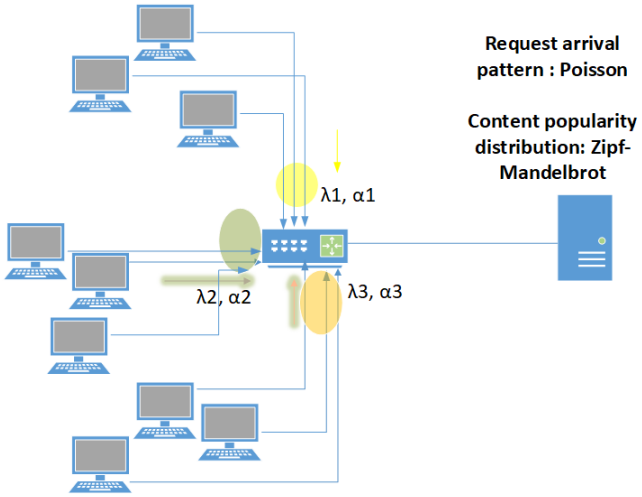

Figure 3. Illustration of the demand model.

Contents on the network are divided into several classes according to their characteristics [17]. The content class classification is based on delay and jitter requirements, content usage duration, and the request rate pattern at a certain interval for each class. The first class requires low delay and low jitter with the contents repeatedly requested by many users in years, and the request rate is medium relative to other classes. The second class is User Generated Content (UGC), with a the medium requirement delay and less jitter. UGC comprises of any content that has been created and published by unpaid contributors, such as blogs, social media content, etc. It has a medium content use duration, and in most situations, it is repeatedly request for a few weeks. Furthermore, it has a high request rate relative to other content classes. The third content class is a web application, which has the same requirement of delay and jitter as the second class, and a higher usage period than the first and second class. The first class has the highest priority in terms of the delay and jitter requirements, followed by the second and third classes. Table 1 shows that there are four content service classes with three service classes that need to be cached, while one does not require cache system. The Cache Based on Popularity and Class (CAPIC) algorithm pays attention to content classification procedure to determines the amount of content thatcan be stored on the content store.

Table 1. Class of content services.

Class	Example	Delay req.	Jitter req.	Duration	Request Rate
1	Real time/VoD	Low	Low	Long (years)	Low-Mid
2	User Generated Content (UGC)	Mid	No	Mid (weeks)	Mid
3	Web access	Mid	No	Mid (weeks)	High
4	email, VoNDN	No	No	Short(days)	Not cached

In this research, the number of requests for a content service class follows the Poisson distribution. The Zipf–Mandelbrot distribution, which contains a flattened factor, is used to model requests for data due to content popularity [24–26]. Content on the top rankings has a greater probability of being requested by consumers compared to lower rankings. The probability of consumers accessing content i is expressed by (1):

$$p(i) = \frac{(i+q)^{-\alpha}}{\sum_{i=1}^{N}(i+q)^{-\alpha}} \quad (1)$$

where:

$p(i)$ = probability request of content i,
N = number of files or content,
q = flattened factor,
α = exponential factor.

3. Static-CAPIC: Cache Based on Popularity and Class with the Static Cache Proportion

The purpose of Static-CAPIC is to increase the network cache hit ratio using its simple mechanism while considering the popularity and content class. The basic idea is to keep the favorite content as close as possible to the consumer, by dividing the routers' content store for each class and assigning the different cache capacity for each content class according to their needs and character. The traffic character for each content class is modeled based on Table 1. The determination of cache capacity can be carried out by observing the optimum conditions. Furthermore, the Static-CAPIC performs a two-steps CAPIC mechanism as explained in Section 2.2, where the proportion of storage for each content class is pre-determined statically. Once the proportion is set, it does not change.

Mathematical Analysis for Static-CAPIC

In this research, the term 'cache hit' is used to denote the number of requests for content that can be responded to by routers, and without asking it to the producer. Therefore, the router's ability to serve the request makes it more efficient. The term 'missed request' is used to determine the number of requests that cannot be responded by the router. Meanwhile, the 'cache hit ratio' is a number of content requests in a cache that is successfully fulfilled, compared to the many requests it receives. The Cache Hit Ratio, CHR, for content stores without partitions is obtained using the following formula:

$$CHR = \frac{H}{H + M} \tag{2}$$

where:

H = number of hit requests,
M = number of missed requests.

The commonly used cache replacement techniques for NDN are Least Frequently Used (LFU) and Least Recently Used (LRU). According to a research carried out by Jing et al. [3], LFU is the most effective cache replacement strategy for the network with request patterns following Zipf–Mandelbrot distribution. LFU is resilient to changes in the demand distribution (exponential factor in the Zipf distribution), while LRU responds poorly to Zipf's flatter distribution. The benefits of LRU are significant at higher values of the exponential factor, therefore this research discusses, analyzes, and elaborates on the content store conditions with and without partitioning using the cache replacement Least Frequently Used (LFU) technique.

The research used parameter C as the size of content store in each router. To make it easier to analyze, C is used to determine the maximum number of the packet that can be stored in the content store. In the simple case of a system with content stores that only have a single content class and use an LFU algorithm, C number of content can be stored according to the content's popularity. Content store with a size C can cache the content ranking into $1, 2, 3, ..., C$. The hit probability is the probability of content that is requested available on the content store. The content that is cached in the LFU content store is according to the top ranking in popularity, and $p(i)$ indicates the probability content i is requested by consumers, it can be written that the hit probability of content 1 is relatively the same as $p(1)$, the hit probability of content 2 is relatively the same as $p(2)$, the hit probability of content C is relatively the same as $p(C)$.

From (1) and (2), we can write cache hit ratio in content store with size C:

$$CHR(C) = I \cdot \frac{p(1)+p(2)+p(3)+...+p(c)}{I}$$
$$CHR(C) = p(1) + p(2) + p(3) + \ldots + p(C)$$

$$CHR(C) = \sum_{i=1}^{C} p(i) \tag{3}$$

where

C = size of the content store,
$p(i)$ = request probability for content i,
I = total number of interest from consumer.

In the case of the content store with partitions, and according to content class, it is assumed that there are K service classes. Therefore there are K partitions in the content store. This is in accordance with Figure 4, with each class having different request characteristics according to Table 1. Assuming a service of class k has a different average rate request, λ_k with $k = (1, 2, \ldots, K)$, the cache hit ratio for the content store which is partitioned into class k with the cache size c_k, the pattern of request arrival per time unit follows the Poisson distribution with rates λ_k and request patterns follow Zipf–Mandelbrot with content access probabilities i, $p(i)$, can be written as:

$$CHR(C) = \frac{\sum_{i=1}^{c1} (p(i) \,.\, \lambda_1) + \ldots + \sum_{i=1}^{ck} (p(i) \,.\, \lambda_k)}{\sum_{i=1}^{k} \lambda_k} \tag{4}$$

and

$$c_1 + c_2 + \ldots + c_k = C$$

where:

k = (1, 2, 3, …, K) = content class,
$p(i)$ = hit probability for content i,
c_k = cache size for content class k,
λ_k = average number of request per time unit, >0,
C = total size of content store.

Figure 4. Illustration of K number of class partitions on the content store.

When the average request rate for each content class is stated as $\lambda_1 < \lambda_2 < \lambda_3$, and each content class has an identical Zipf–Mandelbrot distribution pattern, the base on the Equation (4), is used to get the largest $CHR(C)$ value. Furthermore, the class with the largest average interest rate, λ_k, needs to be given the biggest proportion of cache c_k, followed by a smaller proportion for classes with smaller average interest arrivals. However, consumers demand can change at any time. For instance, they can ask for specific content in one class more than the other, which changes with time. The number of consumers' requests fluctuates in a limited time, and in a long period, it produces an average value of demand following Poisson distribution. This condition needs to be accounted for to provide better performance. Therefore, based on this mathematical analysis, for Static-CAPIC, the cache proportion is determined in accordance with the approximate consumer interest rate. Naturally,

the class with a higher interest rate needs more cache proportion to acquire the high cache hit ratio. The simulation result shown in Section 5.2 indicates that the network cache hit ratio is more significant than the LCD+sharing scheme, which means that the goal of Static-CAPIC has been reached. However, in detail, the first class has the smaller cache hit ratio compared to the sharing scheme, whereas it has to be prioritized compared to other class due to delay and jitter requirement. This result will be explained in more detail in Section 5.2.

4. Dynamic-CAPIC: Cache Based on Popularity and Class with Dynamic Cache Proportion

In the Static-CAPIC described earlier, the proportions are statically and easily pre-determined for known and fixed traffic patterns. However, this is not optimal because the demand characteristic fluctuates at intervals. Although Static-CAPIC has succeeded in increasing the network cache hit ratio, it has not delivered the expected cache hit ratio for the first class of content. For this reason, Static-CAPIC is developed into Dynamic-CAPIC that aims to optimize the cache hit ratio, accommodate the real-time changing request, and give each content class the appropriate cache hit ratio. Figure 5 shows that each content class's cache proportion is the same as all the intervals for Static-CAPIC. Meanwhile, in Dynamic-CAPIC, the cache size is changed dynamically based on the actual demand pattern of consumers.

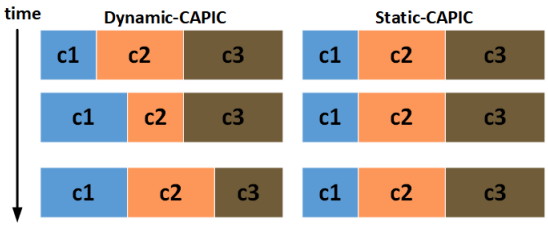

Figure 5. Dynamic-CAPIC vs. Static-CAPIC.

4.1. Formulation of Determining the Cache Proportion

To optimize the network cache hit ratio, the cache size for each content class needs to be specified according to its needs and character. In addition, a caching mechanism capable of providing optimal performance for all content classes and also for the entire system is needed.

Equation (4) shows that generally, the cache hit ratio is affected by the number of actual requests for the content and the content store's capacity. Therefore, the following mathematical equation is used to determine the important factors for the cache portion of every content class. In this system, three content classes are modeled based on Table 1, which is followed by the use of the Lagrange Multiplier. Based on Equation (4), the cache hit ratio function, called the function f, can be written as follows:

$$f(c_1,\ c_2,\ c_3) \ = \frac{p(1)\cdot\lambda_1+p(2)\cdot\lambda_1+..+p(c_1)\cdot\ \lambda_1+p(1)\cdot\lambda_2+p(2)\cdot\lambda_2+..+p(c_2)\cdot\lambda_2+p(1)\cdot\lambda_3+p(2)\cdot\lambda_3+..+p(c_3)\ \lambda_2}{\lambda_1+\lambda_2+\lambda_3}$$

and as in Equation (1),

$$p\ (i) = \frac{(i+q)^{-\alpha}}{\sum_{i=1}^{N}(i+q)^{-\alpha}}$$

The constraint function is written as the function g,

$$g(c_1,\ c_2,\ c_3\ ) = c_1\ +\ c_2\ +\ c_3\ = C$$

then the Lagrange function is:

$$\nabla f(c_1,\ c_2,\ c_3\ ) = \ L\ \cdot \nabla g(c_1,\ c_2,\ c_3\ )$$

where:

L = Lagrange multiplier,
λ_k = request arrival rate for class k,
c_k = cache proportion for class k.

The first derivative of the function f with c_1, c_2, c_3 is:

$$\frac{\partial f}{\partial c_1} = \frac{-\alpha(c_1+q)^{-\alpha-1}}{\sum_{i=1}^{N_1}(i+q)^{-\alpha}} \cdot \lambda_1$$
$$\frac{\partial f}{\partial c_2} = \frac{-\alpha(c_2+q)^{-\alpha-1}}{\sum_{i=1}^{N_2}(i+q)^{-\alpha}} \cdot \lambda_2$$
$$\frac{\partial f}{\partial c_3} = \frac{-\alpha(c_3+q)^{-\alpha-1}}{\sum_{i=1}^{N_3}(i+q)^{-\alpha}} \cdot \lambda_3$$

So that we can write the Lagrange equation for each class:

$$L = \frac{-\alpha\,(c_1+q)^{-\alpha-1}}{\sum_{i=1}^{N_1}(i+q)^{-\alpha}} \cdot \lambda_1 \tag{5}$$

$$L = \frac{-\alpha\,(c_2+q)^{-\alpha-1}}{\sum_{i=1}^{N_2}(i+q)^{-\alpha}} \cdot \lambda_2 \tag{6}$$

$$L = \frac{-\alpha\,(c_3+q)^{-\alpha-1}}{\sum_{i=1}^{N_3}(i+q)^{-\alpha}} \cdot \lambda_3 \tag{7}$$

From Equations (5) and (6) can be obtained:

$$\frac{-\alpha\lambda_1\,(c_1+q)^{-\alpha-1}}{\sum_{i=1}^{N_1}(i+q)^{-\alpha}} = \frac{-\alpha\lambda_2\,(c_2+q)^{-\alpha-1}}{\sum_{i=1}^{N_2}(i+q)^{-\alpha}} \tag{8}$$

To simplify, assumed a value $q = 0$,

$$c_2 = c_1 \cdot \sqrt[\alpha+1]{\frac{\lambda_2 \sum_{i=1}^{N_1}(i)^{-\alpha}}{\lambda_1 \sum_{i=1}^{N_2}(i)^{-\alpha}}}$$

In the same way, from Equations (5) and (7), we obtain:

$$c_3 = c_1 \cdot \sqrt[\alpha+1]{\frac{\lambda_3 \sum_{i=1}^{N_1}(i)^{-\alpha}}{\lambda_1 \sum_{i=1}^{N_3}(i)^{-\alpha}}}$$

and from Equations (6) and (7) we get:

$$c_3 = c_2 \cdot \sqrt[\alpha+1]{\frac{\lambda_3 \sum_{i=1}^{N_2}(i)^{-\alpha}}{\lambda_2 \sum_{i=1}^{N_3}(i)^{-\alpha}}}$$

By connecting it to the constraint equation, it can be written down:

$$c_1 + c_2 + c_3 = C$$

$$c_1 = \frac{C}{\left(1 + \sqrt[\alpha+1]{\frac{\lambda_2 \sum_{i=1}^{N_1}(i)^{-\alpha}}{\lambda_1 \sum_{i=1}^{N_2}(i)^{-\alpha}}} + \sqrt[\alpha+1]{\frac{\lambda_3 \sum_{i=1}^{N_1}(i)^{-\alpha}}{\lambda_1 \sum_{i=1}^{N_3}(i)^{-\alpha}}}\right)} \tag{9}$$

$$c_2 = \frac{C}{\left(\frac{1}{\sqrt[\alpha+1]{\frac{\lambda_2 \sum_{i=1}^{N_1}(i)^{-\alpha}}{\lambda_1 \sum_{i=1}^{N_2}(i)^{-\alpha}}}} + 1 + \sqrt[\alpha+1]{\frac{\lambda_3 \sum_{i=1}^{N_2}(i)^{-\alpha}}{\lambda_2 \sum_{i=1}^{N_3}(i)^{-\alpha}}} \right)} \tag{10}$$

$$c_3 = \frac{C}{\left(\frac{1}{\sqrt[\alpha+1]{\frac{\lambda_3 \sum_{i=1}^{N_1}(i)^{-\alpha}}{\lambda_1 \sum_{i=1}^{N_3}(i)^{-\alpha}}}} + \frac{1}{\sqrt[\alpha+1]{\frac{\lambda_3 \sum_{i=1}^{N_2}(i)^{-\alpha}}{\lambda_2 \sum_{i=1}^{N_3}(i)^{-\alpha}}}} + 1 \right)} \tag{11}$$

Based on Equations (9)–(11), it can be concluded that to maximize the cache hit ratio, the proportion of cache given for each class k, c_k, has to be directly proportional to the total content store's size, requests rate, and document population number of its class. The given cache proportion is inversely proportional to other content class's document population number and request rate. This is in accordance with the content population number N_1, N_2 and N_3. NDN routers can not know λ_1, λ_2 and λ_3 in real-time, therefore an additional parameter is used to describe these two conditions in real-time. Due to the limitation of the content store's capacity, all the content cannot be cached, therefore, the important ones are selected and stored. Dynamic-CAPIC considers this in the second step of CAPIC, as described in Section 2.2, which determines how much content is worth keeping, in accordance with a certain frequency. For this reason, this research proposes a formula for determining the proportion of storage for k content classes in Dynamic-CAPIC. This formula includes parameter X_k, which represents the frequency of requests and also the amount of variation of content requested in real-time.

$$c_k = \frac{X_k \cdot C}{\sum_{i=1}^{k} X_i} \tag{12}$$

where:

X_k = the number of variations in content class k that is requested more than a certain Frequency Limit Factor value,
k = content class $(k = 1, 2, 3, \dots)$,
c_k = content store proportion for class k,
C = total content store size.

4.2. Analysis of Determining the Content Variation X_k

The Frequency Limit Factor needs to be determined before calculating the proportion of cache using Equation (12). The Frequency Limit Factor F_{Rk} has determined the following three stages:

1. Determine the value of parameter D, which is the percentage of content class that is likely to be considered in the system (e.g., 10%, 20%, etc.).
2. Calculate parameter R, using N_k as a total number of content in class k

$$R = D \times N_k \tag{13}$$

3. Calculate the Frequency Limit Factor F_{Rk},

$$F_{Rk} = p(R) \times \lambda_k \tag{14}$$

These three initial stages can be carried out by analyzing general patterns of traffic on a network with classes 1, 2, or 3 used as a reference. However, for the case $\lambda_1 < \lambda_2 < \lambda_3$, the value of

$F_{R1} < F_{R2} < F_{R3}$. For example, classes 1, 2, and 3 have the amount of content 500, 700, and 1000 and their demands are 10, 20, and 30 $request/time - unit$. In case we decided to use D = 30%, the parameter R can be calculated as follows:

$$R = D \cdot N_1 = 30\% \cdot 500 = 150$$

The Frequency Limit Factor that will be used in the system is calculated following the first class reference ($k = 1$)

$$F_{Rk} = p(R) \times \lambda_k$$
$$F_{R1} = p(150) \times \lambda_1$$

Figure 6 is a graphical representation of the explanation for clarity. The red arrow points to the rank of content, called parameter R, which is 150. The horizontal black line points the Frequency Limit Factor, F_{R1}. Therefore, to get X_k for each class, the number of content which is requested more than the Frequency Limit Factor is checked. The value of X_k for $k = 1, 2$ and 3 can be written as:

$$X_1 = 150 \text{ (indicated by the red arrow)}$$
$$X_2 = 309 \text{ (indicated by the green arrow)}$$
$$X_3 = 513 \text{ (indicated by the blue arrow)}$$

Thus, the proportion of each class can be calculated (for $k = 1, 2, 3$) using Equation (12). In the case of a total content store size of 100, it can be defined that:

$$c_1 = \frac{150}{150+309+513} \times 100 \approx 15$$
$$c_2 = \frac{309}{150+309+513} \times 100 \approx 32$$
$$c_3 = \frac{513}{150+309+513} \times 100 \approx 53$$

The value of X_k can change dynamically and in real-time. This formula can be used even if the system (or router) does not know the total amount of content for each class.

Unlike the Static-CAPIC algorithm, in Dynamic-CAPIC a formula is used to dynamically calculate the proportion of cache for each content. One of the parameters used is the Frequency Limit Factor which determines the amount of top content needed to calculate the proportion of cache in the content store for each class. For example, as an illustration, the communication system has the total frequency of request 10,000. Figure 7 shows that in first class (red line) the number of top content that has many requests greater than Frequency Limit Factor of 25, is approximately 87 contents. For a Frequency Limit Factor of 75, the number of top content calculated is approximately 21.

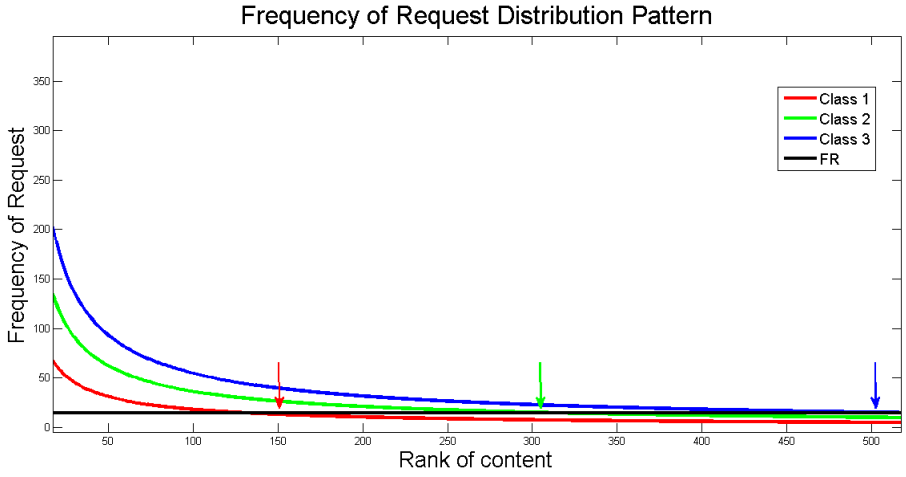

Figure 6. Illustration of Request Distribution Pattern and Frequency Limit Factor.

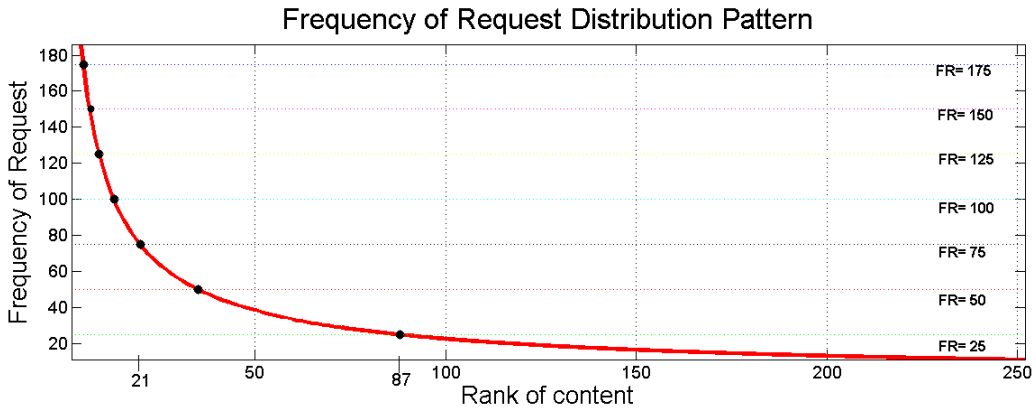

Figure 7. Number of top content for 1st class with various Frequency Limit Factors.

5. Simulation Models and Results

This research built a comprehensive simulation environment using Matlab to analyze the performance of Static-CAPIC and Dynamic-CAPIC. Furthermore, it also compares both techniques with the combination of two existing techniques, LCD+sharing scheme. The network uses best path routing as a common routing algorithm in NDN. The LCD+sharing scheme carried out the LCD scheme as a first step. Whenever consumers request the content, the router sends data to the consumer and store its copy in the one-hop downstream router. There is no partition in the content store. Every content class is cached as long as it does not exist in the content store when requested, and the content store is not full yet. However, when the content store is full, a cache replacement mechanism is obtained.

The network containing the consumer nodes, producers, and router nodes was modeled in this study to evaluate Static-CAPIC and Dynamic-CAPIC schemes performance. Each router node has a content store with the same total capacity. Contents on the network are grouped into different classes, as in Table 1. For Static-CAPIC, each class has a statically pre-determined proportion in each router. In contrast, for Dynamic-CAPIC, it is changing dynamically using the Dynamic CAPIC formula as in Equation (12).

The study assumed that the data have the same size. Every request with an interest package is responded with a data/content package, either by routers or producer nodes. These assumptions simplify the model and make it easier to analyze. One slot in the content store is used to save one packet of content. Therefore, the content store's capacity is equal to the number of packets that can be saved.

5.1. Simulation Models

To evaluate the performance of Static-CAPIC and Dynamic-CAPIC compared to the LCD+sharing scheme, a test with three content classes that need to be cached was carried out, according to Table 1. The detail of simulation parameters as shown in Table 2. For the Static-CAPIC, a pre-determined cache proportion for the first, second, and third class are 15, 42, and 92 packets, respectively. These values are based on the percentage of content population number for each class, which are 5%, 10%, and 15% of the first, second, and third class content population, normalized to the total content store size, which is 150. The parameters analyzed for these tests are cache hit ratio and average path stretch. The cache hit ratio illustrates the ratio between the number of hit requests and the total of consumer requests. The average path stretch indicates the number of hops from the consumer to the node with the requested data. The bigger cache hit ratio is better, while the more significant path stretch is worse.

Table 2. Simulation parameters.

Parameter	Value
Number of routers	20
Number of class	3
Content store proportion for static-CAPIC (classes 1, 2, and 3)	1. Dynamic-CAPIC = dynamic, according to the formula
	2. Static-CAPIC = 15:43:92 (5%, 10% and 15% of content number, normalized)
	3. LCD+Sharing scheme = sharing
Number of content	500 (1st class);
	700 (2nd class);
	1000 (3rd class)
Number of rounds	10,000
Average request rate per round for each class	10 (1st class)
	20 (2nd class)
	30 (3rd class)
Exp factor	0.8
Flattened factor	3
F_{Rk}	25, 50, 75, 100, 125, 150, 175
Scenario	Simulating the Dynamic-CAPIC vs. Static-CAPIC with a variety of Frequency Limit factor

Consumer's requests are modeled as described in Section 2.3. For Static-CAPIC and Dynamic-CAPIC, each content class gets a different treatment and different cache portion in the content store. This differs from the LCD+sharing scheme, where there is no separation in the content store, and every content class is treated in the same manner.

The greater the Frequency Limit Factor, the more limited the amount of top content considered in determining the proportion of cache in the content store. Setting a too small value of Frequency Limit Factor causes all content in the class to be evaluated to determine the proportion of cache.

5.2. Test Results

The simulation result in Figure 8 (above) shows the first class cache hit ratio, while Figure 8 (below) indicates path stretch of Dynamic-CAPIC, Static-CAPIC, and LCD+sharing scheme. Figure 9 (above) and Figure 10 (above) shows the cache hit ratio while Figure 9 (below) and Figure 10 (below) illustrates the path stretch for second class and third class Dynamic-CAPIC, Static-CAPIC, and LCD+sharing scheme. The network cache hit ratio of Dynamic-CAPIC, Static-CAPIC, and LCD+Sharing scheme are presented in Figure 11. Furthermore, graphs comparing the cache hit ratio, and each content class's path stretch in an internal Dynamic-CAPIC, Static-CAPIC, and LCD+sharing scheme, respectively, are shown in Figures 12–14.

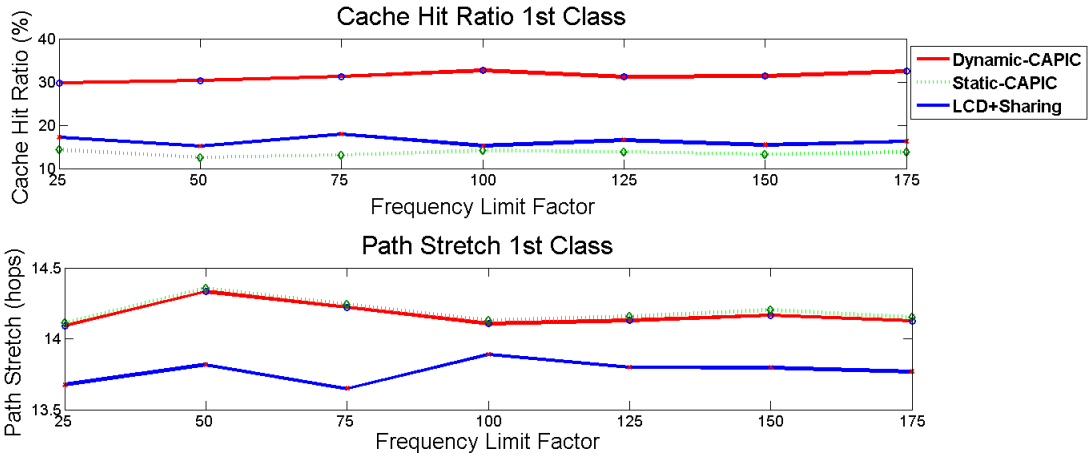

Figure 8. Cache hit ratio (**above**) and path stretch (**below**) 1st class of Dynamic-CAPIC, Static-CAPIC, and LCD+sharing scheme.

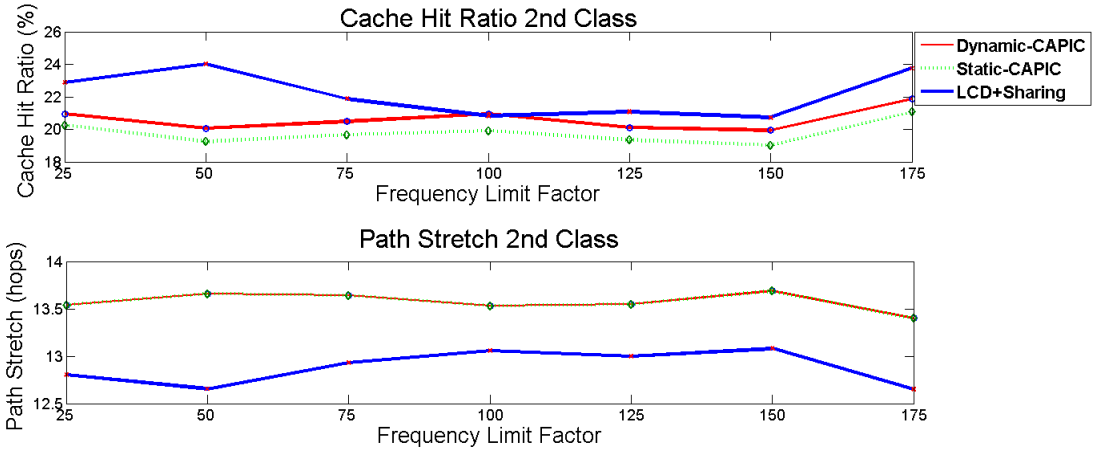

Figure 9. Cache hit ratio (**above**) and path stretch (**below**) 2nd class of Dynamic-CAPIC, Static-CAPIC, and LCD+sharing scheme.

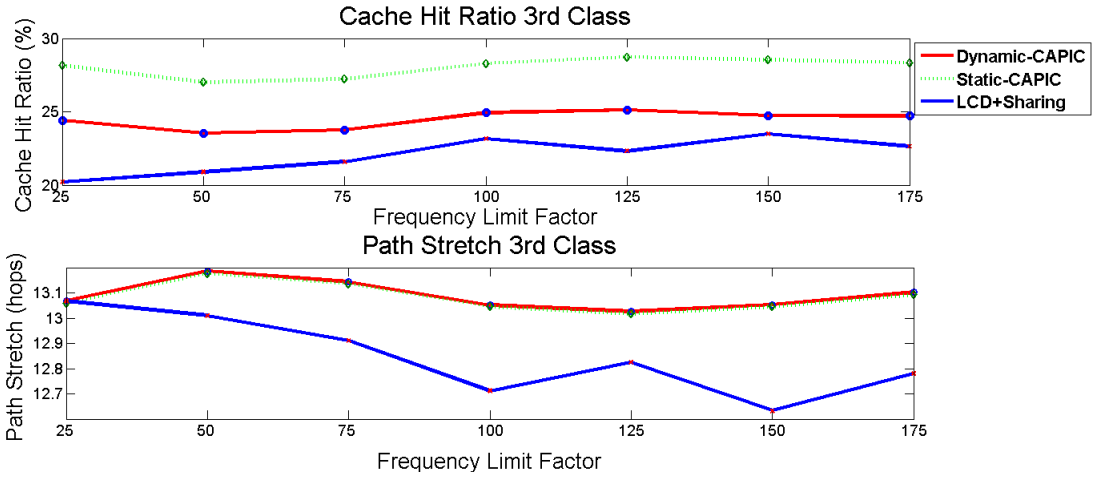

Figure 10. Cache hit ratio (**above**) and path stretch (**below**) 3rd class of Dynamic-CAPIC, Static-CAPIC, and LCD+sharing scheme.

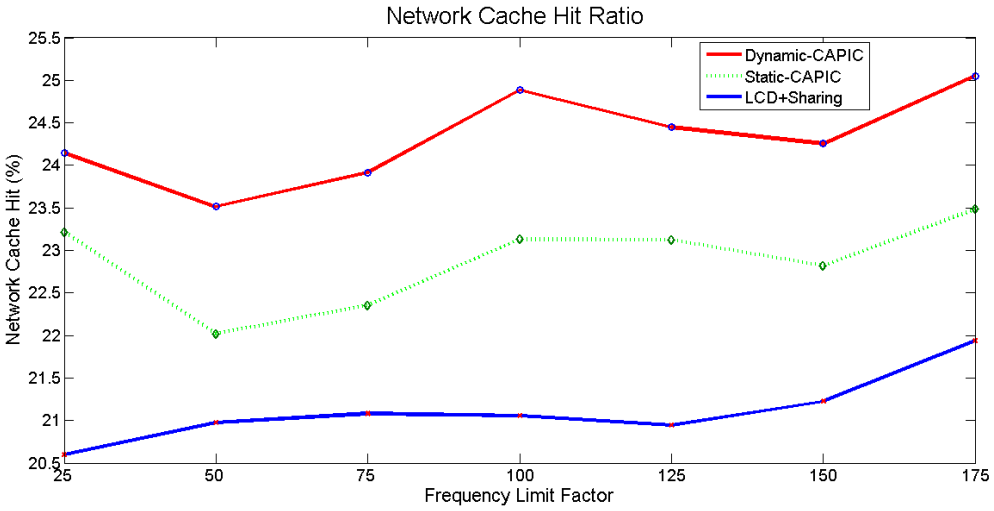

Figure 11. Network cache hit ratio Dynamic-CAPIC, Static-CAPIC, and LCD+sharing scheme.

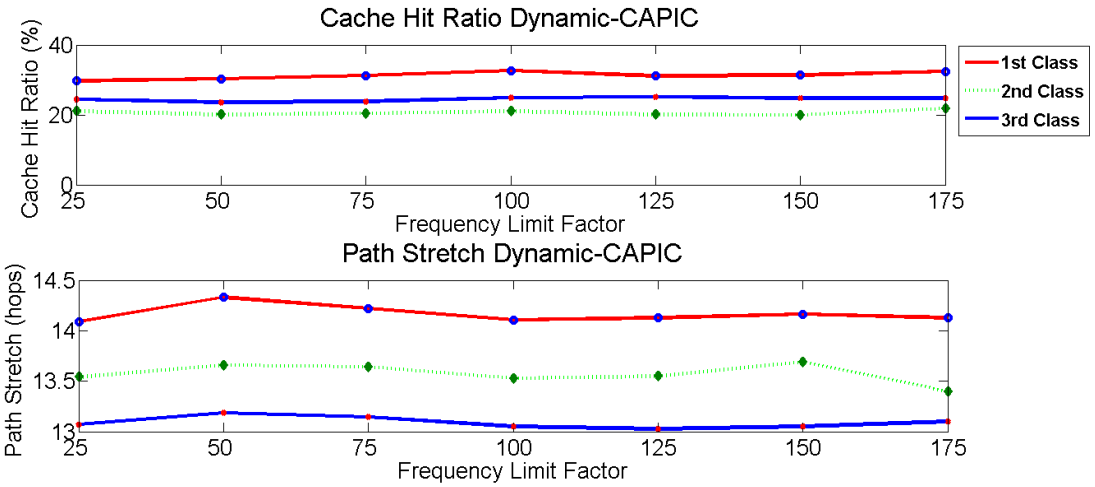

Figure 12. Performance of Dynamic-CAPIC.

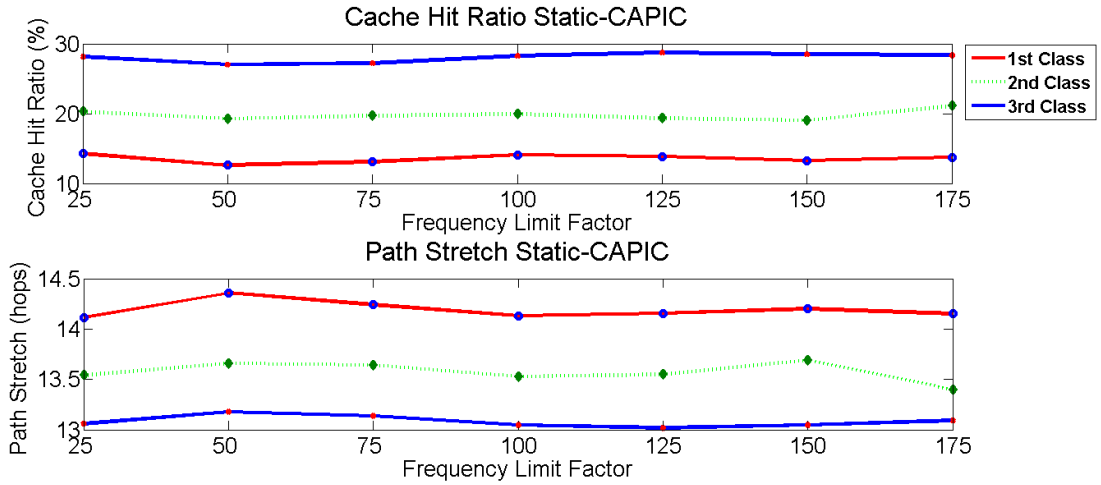

Figure 13. Performance of Static-CAPIC.

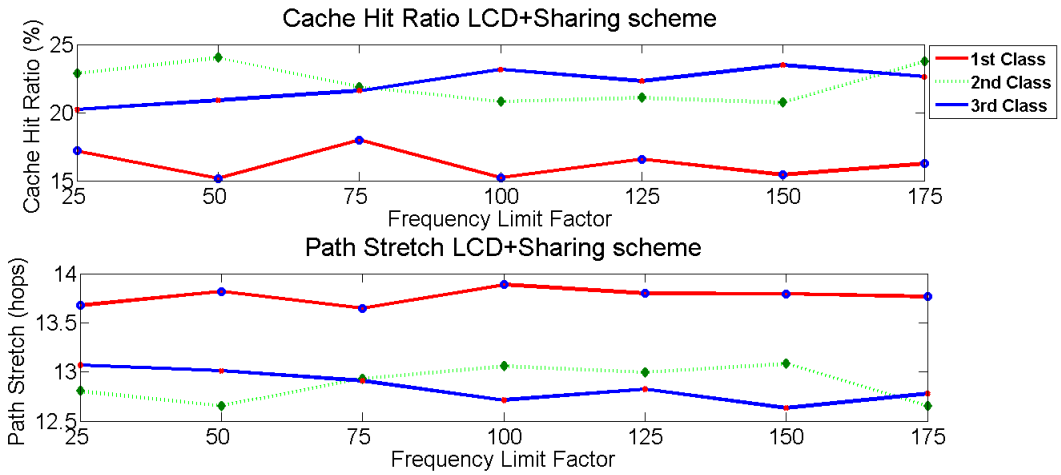

Figure 14. Performance of LCD+sharing scheme.

Dynamic-CAPIC gave the highest cache hit ratio for the first class compared to Static-CAPIC and LCD+sharing scheme. Dynamic-CAPIC gave a 130.7% in an average increase in the cache hit ratio compared to Static-CAPIC, and a 92.4% in an average increase compared to the LCD+sharing scheme. The Dynamic-CAPIC produces the highest cache hit ratio for the first class compared to the second and third classes, are shown in Figure 12. This result is as expected where the first class is concerned with jitter and delay, so content is expected to be available at the nodes in the network without asking the producer, which has the longest distance from the consumer. The condition is different for the Static-CAPIC algorithm. Static-CAPIC gives the largest cache hit ratio to the third class, instead of the first class, as shown in Figure 13. Dynamic-CAPIC corrects this condition. For the path stretch parameter, as in Figure 8 (below), the path stretch of Dynamic-CAPIC is similar to Static-CAPIC.

For the second class, LCD+sharing schemes provide a higher cache hit ratio than Dynamic-CAPIC and Static-CAPIC algorithm, as in Figure 9 (above). The cache hit ratio of LCD+sharing scheme is 6.95% in an average higher compared to Dynamic-CAPIC. Dynamic-CAPIC has a cache hit ratio that is higher than Static-CAPIC by 4.21%. Dynamic-CAPIC cache hit ratio lies between LCD+sharing scheme's and Static-CAPIC's cache hit ratio. Dynamic-CAPIC compromises the cache hit ratio in this second content class to provide a larger cache hit ratio for the first class. For the path stretch parameter, it is similar between Dynamic-CAPIC and Static-CAPIC as in Figure 9 (below).

For the third class, Dynamic-CAPIC gives the cache hit ratio between Static-CAPIC and the LCD+sharing scheme ratio, as shown in Figure 10 (above). Dynamic-CAPIC has a greater cache hit ratio than LCD+sharing scheme of 10.98% on an average and has a smaller cache hit ratio than Static-CAPIC of 12.77%. For the path stretch, as in Figure 10 (below), the path stretch of Dynamic-CAPIC is similar to Static-CAPIC.

Figure 14 shows that by using LCD+sharing scheme, it is challenging to create a priority mechanism for each content class. The first class of LCD+sharing scheme has the lowest cache hit ratio, and it is unstable for the second and third classes. Content that is popular for one class can be deleted due to a lack of resources to store others. This condition will make the network cache hit ratio not optimal for LCD+sharing scheme. Dynamic-CAPIC provides a highest cache hit ratio for the first class, followed by the third and second classes. These results have fit the requirement of each content class of Dynamic-CAPIC. Dynamic-CAPIC also provides the highest network cache hit ratio compared to Static-CAPIC and LCD+sharing scheme. Meanwhile, Static-CAPIC gives a higher network cache hit ratio than the LCD+sharing scheme, but it cannot provide an appropriate cache hit ratio, especially for the first content class.

The consumers' demand traffic distribution is complex. In each internal content class, every content popularity follows the Zipf–Mandelbrot distribution, and the average request rate of each class is modeled by Poisson distribution. At any single time, the number of requests for every content in every content classes can change. The Dynamic-CAPIC mechanism provides the cache proportion's calculation process according to each content class's real-time needs. The cache proportion is changing every time with a content class possessing the biggest cache proportion, sometimes. In another time, this class tends to have the smallest cache proportion. The condition depends on the request from consumers. As a result of this flexibility, looking from the network side as in Figure 11, the Dynamic-CAPIC method outperforms the Static-CAPIC and LCD+sharing scheme because Dynamic-CAPIC can keep up with changes in demand patterns every time. Simultaneously, the number of requests for every content in all content classes changes, while the average number of requests has been defined using the Poisson distribution. Dynamic-CAPIC provides calculation of cache portions according to the real-time needs of each content class. Dynamic-CAPIC's cache hit ratio is the biggest, followed by Static-CAPIC's and LCD+sharing scheme's cache hit ratio for all Frequency Limit Factor uses. Dynamic-CAPIC gives an average cache hit ratio of 6.29% compared to Static-CAPIC and an average of 15.15% compared to LCD+sharing scheme.

This research measure the complexity of the algorithm using the Time Complexity parameter. Time complexity is used to measure the time required to run an algorithm with specific inputs. However, the processing time will depend on the type of device, the parallel process being worked on, network load, etc. The value can be biased. For this reason, time complexity can be demonstrated through Big-O notation. Big-O notation converts all steps of an algorithm into a more objective form of algebra. The pseudocode of Dynamic-CAPIC and LCD+sharing scheme is presented in Algorithm 1.

The Dynamic-CAPIC algorithm is written mathematically as an input correlation as, 4K + 2, and the Big-O notation is O(K). The Big-O notation for the LCD+sharing scheme is O(1) because the complexity is always the same for any input value. Ultimately, the complexity of Dynamic-CAPIC is growing as K increase. In addition to the growth of complexity, Dynamic-CAPIC gives the worth compensation. It provides the highest network's cache hit ratio compared to LCD+sharing scheme for about 8%. The most important thing is Dynamic-CAPIC can control each content class performance following the class requirement in real-time. It gives a maximal 130 % cache hit ratio compared to the LCD+sharing scheme for the first content class.

Algorithm 1: Function Dynamic-CAPIC and Function LCD+Sharing.

```
Initialization;
K: total number of content class ;
k: content class k=[1, 2, . . . , K];
C: total capacity of the content store;
Function DCAPIC(input: C,k,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
% cek in the router;
Calculate x_k based on F_R ;
for m=1: K do
    Total_X =+ x_m;
% calculate cache portion of content class k;
c_k = (X_k · C) / Total_X;
Check capacity of c_k;
if capacity not full then
    save i in part-of-Content-store;
    send i to consumer ;
else
    do replacement algorithm;
    save i in part-of-Content-store;
    send i to consumer ;
Function LCD-Share(input: C,K,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
%cek in the router;
Check Content-store;
if i not-exist in Content-store then
    Check total-capacity-Content-store;
    if capacity not full then
        save i in Content-store;
    else
        do the replacement algorithm;
        save i in Content-store;
        send i to consumer ;
else
    send i to consumer;
    end process;
```

6. Conclusions

In conclusion, this research explains the basic idea of Cache Based on Popularity and Class (CAPIC) with two cache determination proportions, that are static and dynamic. The mathematical formula was derived to analyze and optimize the cache hit ratio for the content store, with the Static-CAPIC described. The Static-CAPIC algorithm fulfills its purpose to improve the total cache hit ratio on the network. It provides a simple cache technique, however, it was not perfect at controlling the cache hit ratio for priority class. The Dynamic-CAPIC, as an improved algorithm of Static-CAPIC, provides flexibility to change the cache proportion based on the frequency of requests in real-time. The formula of Dynamic-CAPIC involves consideration of the consumer request all the time. It gives the appropriate cache hit ratio that is fit for each content class's character. This algorithm also outperforms the Static-CAPIC, and the LCD+sharing scheme in total network cache hit ratio parameter. The average path stretch of the Dynamic-CAPIC is relatively the same compared to the Static-CAPIC scheme.

7. Future Research

For future research, Dynamic-CAPIC will be conducted with the capability to estimate the future demand of content. Furthermore, a wider range of Frequency Limit Factors need to be provided for better performance.

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.K.; writing—original draft, L.V.Y.; validation, N.R.S. and I.J.M.E.; writing—review editing, N.R.S. and N.K.; supervision, N.R.S., I.J.M.E., and N.K.; resources, I.J.M.E. and N.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Ministry of Research Technology and Higher Education Republic of Indonesia through the sandwich-like program (PKPI 2019) for the research collaboration in Japan, and Doctoral Dissertation Research grant (PDD 2019) for research publication.

Acknowledgments: This research was partially supported by Telkom University, and Ministry of Research Technology and Higher Education Republic of Indonesia through the sandwich-like program (PKPI 2019) for the research collaboration in Japan, grant number B/865/D3.2/KD.02.00/2019. The publication of research is supported by Doctoral Dissertation Research grant (PDD 2019), grant number 2/E1/KP.PTNBH/2020.

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
CAPIC	Cache Based on Popularity and Class
FIB	Forwarding Information Base
PIT	Pending Interest Table
LCD	Leave Copy Down
UGC	User Generated Content
VoD	Video on Demand
TTL	Time to Live

References

1. Choi, J.; Han, J.; Cho, E.; Kwon, T.T.; Choi, Y. A survey on content-oriented networking for efficient content delivery. *IEEE Commun. Mag.* **2011**, *49*, 121–127. [CrossRef]
2. Kamiyama, N.; Murata, M. Dispersing Content Over Networks in Information-Centric Networking. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 521–534. [CrossRef]
3. Jing, Y. *Evaluating Caching Mechanisms In Future Internet Architectures*; Technical Report; Massachusetts Institute of Technology: Cambridge, MA, USA, 2016.

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**, *2017*, 1–32. [CrossRef]
5. Saxena, D.; Raychoudhury, V.; Suri, N.; Becker, C.; Cao, J. ScienceDirect Named Data Networking: A survey. *Comput. Sci. Rev.* **2016**, *19*, 15–55. [CrossRef]
6. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.; Briggs, N.; Braynard, R. Networking named content. *Commun. ACM* **2012**, *55*, 117–124. [CrossRef]
7. Yovita, L.V.; Syambas, N.R.; Yosef, I.E.M. The Effect of Cache Partitioning and Sharing on Named Data Network. *Internetwork. Indones. J.* **2019**, *11*, 87–92.
8. Chu, W.; Dehghan, M.; Towsley, D.; Zhang, Z.L. On allocating cache resources to content providers. In Proceedings of the 2016 3rd ACM Conference on Information-Centric Networking (ACM-ICN 2016), Kyoto, Japan, 26–28 September 2016; pp. 154–159. [CrossRef]
9. Zhang, W.; Fan, R.; Liu, F.; Lai, P. Energy-Aware Caching. In Proceedings of the IEEE 21st International Conference on Parallel and Distributed Systems, Melbourne, Australia, 14–17 December 2015; pp. 473–480. [CrossRef]
10. Zheng, X.; Wang, G. A Cache Placement Strategy with Energy Consumption Optimization in Information-Centric Networking. *Future Internet* **2019**, *11*, 64. [CrossRef]
11. Yovita, L.V.; Syambas, N.R. Caching on named data network: A survey and future research. *Int. J. Electr. Comput. Eng.* **2018**, *8*, 4456–4466. [CrossRef]
12. Yusung, K.; Younghoon, K.; Bi, J.; Yeom, I. Differentiated forwarding and caching in named-data networking. *J. Netw. Comput. Appl.* **2016**, *60*, 155–169. [CrossRef]
13. Wu, T.Y.; Lee, W.T.; Duan, C.Y.; Wu, Y.W. Data lifetime enhancement for improving Qos in NDN. *Procedia Comput. Sci.* **2014**, *32*, 69–76. [CrossRef]
14. Zhang, R.; Liu, J.; Xie, R.; Huang, T.; Yu, F.R.; Liu, Y. Service-aware optimal caching placement for named data networking. *Comput. Netw.* **2020**, *174*. [CrossRef]
15. Carofiglio, G.; Gehlen, V.; Perino, D. Experimental evaluation of memory management in content-centric networking. In Proceedings of the IEEE International Conference on Communications, Kyoto, Japan, 5–9 June 2011; pp. 1–6. [CrossRef]
16. Cho, K.; Lee, M.; Park, K.; Kwon, T.T.; Choi, Y.; Pack, S. WAVE: Popularity-based and collaborative in-network caching for content-oriented networks. In Proceedings of the IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 April 2012; pp. 316–321. [CrossRef]
17. Sandvine Intelligent Broadband Network. *Identifying and Measuring Internet Traffic: Techniques and Considerations*; An Industry Whitepaper; Sandvine Intelligent Broadband Network: Fremont, CA, USA, 2015; pp. 1–20.
18. Cantor, L. Phenomena Report. Technical Report. February 2020. Available online: http://www.fullertreacymoney.com/system/data/files/PDFs/2020/May/COVID%20Internet%20Phenomena%20Report%2020200507.pdf (accessed on 9 December 2020).
19. Khelifi, H.; Luo, S.; Nour, B.; Moungla, H.; Faheem, Y.; Hussain, R.; Ksentini, A. Named Data Networking in Vehicular Ad hoc Networks: State-of-the-Art and Challenges. *Commun. Surv. Tutor.* **2020**, *22*, 320–351. [CrossRef]
20. Yovita, L.V.; Syambas, N.R.; Yosef, I.; Edward, M. 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), Bali, Indonesia, 5–7 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 24–29. [CrossRef]
21. Zhang, M.; Luo, H.; Zhang, H. A survey of caching mechanisms in information-centric networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1473–1499. [CrossRef]
22. Fricker, C.; Robert, P.; Roberts, J.; Sbihi, N. Impact of traffic mix on caching performance in a content-centric network. In Proceedings of the IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 April 2012; pp. 310–315. [CrossRef]
23. Pentikousis, K.; Ohlman, B.; Davies, E.; Spirou, S.; Boggia, G. Information-Centric Networking: Evaluation and Security Considerations, 2016. Available online: https://tools.ietf.org/html/draft-irtf-icnrg-evaluation-methodology-05 (accessed on 9 December 2020).
24. Che, H.; Wang, Z.; Tung, Y. Analysis and design of hierarchical Web Caching System. In Proceedings of the IEEE INFOCOM 2001, Anchorage, AK, USA, 22–26 April 2001; pp. 1416–1424. [CrossRef]

25. Koplenig, A. Using the parameters of the Zipf-Mandelbrot law to measure diachronic lexical, syntactical and stylistic changes—A large-scale corpus analysis. *Corpus Linguist. Linguist. Theory* **2015**, *14*. [CrossRef]
26. Hefeeda, M.; Saleh, O. Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems. *IEEE/ACM Trans. Netw.* **2008**, *16*, 1447–1460.