Article

Compound Popular Content Caching Strategy in Named Data Networking

Muhammad Ali Naeem 1*, Shahrudin Awang Nor 1, Suhaidi Hassan 1 and Byung-Seo Kim 2,*

1 InterNetworks Research Laboratory, School of Computing, Universiti Utara Malaysia, Sintok 06010, Malaysia
2 Department of Software and Communications Engineering, Hongik University, Sejong 30016, Korea
*
Correspondence: jsnbs@hongik.ac.kr

Received: 31 May 2019; Accepted: 8 July 2019; Published: 10 July 2019

Abstract: The aim of named data networking (NDN) is to develop an efficient data dissemination approach by implementing a cache module within the network. Caching is one of the most prominent modules of NDN that significantly enhances the Internet architecture. NDN-cache can reduce the expected flood of global data traffic by providing cache storage at intermediate nodes for transmitted contents, making data broadcasting in an efficient way. It also reduces the content delivery time by caching popular content close to consumers. In this study, a new content caching mechanism named the compound popular content caching strategy (CPCCS) is proposed for efficient content dissemination and its performance is measured in terms of cache hit ratio, content diversity, and stretch. The CPCCS is extensively and comparatively studied with other NDN-based caching strategies, such as max-gain in-network caching (MAGIC), WAVE popularity-based caching strategy, hop-based probabilistic caching (HPC), LeafPopDown, most popular cache (MPC), cache capacity aware caching (CCAC), and ProbCache through simulations. The results show that the CPCCS performs better in terms of the cache hit ratio, content diversity ratio, and stretch ratio than all other strategies.

Keywords: named data networking (NDN); most popular cache (MPC); video on demand (VoD)

1. Introduction

The Internet plays an important role in our lives [1]. It was developed to meet the requirements of an information communication system [2]. Moreover, it delivers social, personal, and academic benefits to its consumers. According to previous forecasts, Internet traffic has grown rapidly in the past several years. A report published by the virtual networking index (VNI) [2] in 2018 found that global Internet traffic increased eight times in the past few years, and the compound annual growth rate (CAGR) was expected to be 35% in the period of 2015–2020 [2,3]. Most Internet traffic is related to content retrieval applications, such as file sharing and YouTube. It is expected that video traffic alone will comprise 86% of all IP traffic in 2019 [4]. Therefore, requirements are increasing for user generated content (UGC) [5] and high-definition video on demand (VoD) traffic [6,7]. In fact, the Internet faced several challenges related to data volume and network traffic, which were caused by the huge consumer requirements for controlling media-driven data content [8].

The Internet has become an essential part of modern life, and Internet traffic is rapidly increasing to meet the exponentially increasing requirements of consumers. Moreover, the number of connected devices is nearly equivalent to one device per-capita and will reach three devices per-capita by 2021 [9,10]. The Internet is based on an outdated communication paradigm, according to which one server sends one title to millions of consumers over the Internet at the same time, meaning that the same data is transmitted many times. The communication architecture delivered by the current host-based Internet will be ineffective to meet future requirements [11,12]. However, named data networking (NDN) [13] is a new Internet architecture that has the potential to handle the current issues...
and is expected to satisfy future requirements to enhance the quality of the Internet. NDN is equipped with in-network cache storage, which is used to store the disseminated content. In the last few years, in-network caching was developed by engineers to modify the design of the Internet [14,15]. Moreover, it has the ability to improve overall data communication services by providing unicast, as well as multicast, data transmissions. In addition, caching facilitates content transmission through low network congestion and flexible bandwidth [16]. The caching is managed by cache management mechanisms.

Nevertheless, NDN faces some limitations in that it needs to establish millions of names for transmitting content and NDN performed poor as compared to plain Hypertext Transfer Protocol (HTTP) and Squid caches [17]. However, NDN is still in its early stage and has lots of advantages for diverse fields. It delivers several benefits for Internet technologies such as the internet of things (IoT) [18], edge cloud computing [19], blockchain [20], distributed fog computing [21], software defined networking (SDN) [22], and fifth generation (5G) mobile cellular networks [23]. These technologies can be integrated with on-path caching to enhance their architectural design to implement the most efficient, flexible, and scalable network services. In the current Internet, the number of data items is significantly large when compared to the number of IP addresses, which implies that it is challenging for the IP address-based architecture to disseminate the desired data in time [24,25]. The on-path caching in NDN can resolve the problems arising in the IP-based Internet by implementing the cache storage within the network nodes. This technique is useful to control the network congestion by caching the disseminated data items at intermediate locations [26]. Therefore, low network congestion results in lower cost and reduction in the response latency. Consequently, it achieves easy control of data traffic. Previously, it was difficult to handle genome data sets but now NDN provides services to handle these types of data sets using NDN caching mechanisms [27,28]. Consequently, the proposed caching strategy will improve the data dissemination and it can emerge with other network technologies (5G, cloud computing, fog computing, and SDN) to get efficient data distribution. In SDN [29], controllers have to be set up reactively or proactively to reduce the flow level forwarding state in open flow switches. Once a setup is built for the flow forwarding state, the SDN does not permit it to restart for subsequent interests, which increases the latency and causes the dissemination process to remain incomplete in the given time duration. In SDN, the failure of the controller can be disturbed to the specific flow of the network because the controller is in charge of all the configurations, operations, and validations of the network resources and topologies. Moreover, it is unsafe for those environments where only one controller is performing because if the controller breaks down, the network flow will be stopped completely as there is no available backup facility [30,31]. Therefore, on-path caching is useful to manage these critical problems and it can deliver better performance by caching the transmitted content at intermediate locations for subsequent interests [32]. Moreover, SDN can be integrated with on-path caching to create a more beneficial network architecture in which the controller of SDN will be managed by NDN [33]. Consequently, the network will be managed through the SDN control plane in which protocols and diverse policies will be merged. The contents within the caching nodes will be controlled by using the data plane and the network devices such as the routers and switches, which will work together on the data plane.

A number of cache management mechanisms were developed, such as max-gain in-network caching (MAGIC) [34], WAVE popularity-based and collaborative caching strategy [35], hop-based probabilistic caching (HPC) [36], LeafPopDown [37], most popular cache (MPC) [38], cache capacity aware caching (CCAC), and ProbCache [39]. However, this study proposes a new cache management strategy called the compound popular content caching strategy (CPCCS). The CPCCS enhances the content selection and content caching mechanisms. Both mechanisms improve the caching performance in terms of cache hit and stretch ratios. Moreover, the CPCCS enhances the content diversity ratio through the selection of heterogeneous contents for caching.
The current study addresses cache management issues in the existing caching mechanisms and proposes a new content caching mechanism to improve the content dissemination through the selection of suitable caching locations for diverse popular content. The rest of the article is organized as follows: In Section 2, max-gain in-network caching (MAGIC), WAVE popularity-based caching strategy, hop-based probabilistic caching (HPC), LeafPopDown, most popular cache (MPC), and cache capacity aware caching (CCAC) are explained as related studies. Section 3 provides critical analysis of related studies. In Section 4, the proposed caching strategy is explained. Section 5 provides the comparative performance of the proposed and benchmark caching strategies using the SocialCCNSim-Master simulator [40], which was specially designed to evaluate the caching performance in NDN. Finally, in Section 6, we conclude the article. In Section 7, the NDN caching is represented as a future perspective.

2. Related Study

The management of the cache depends on how the cache is deployed and managed by the network. The effective management of the cache demonstrates a mechanism that places a copy of the transmitted content into the network caching system efficiently [14]. This mechanism reduces the server stretch length for subsequent interests. Further, it increases the cache hit rate and availability of data [15]. Therefore, the intention of on-path caching is to reduce the usage of the resources of the Internet. For this purpose, on-path caching delivers several cache management strategies that help in overcoming the data transmission problems such as content redundancy and path length between the publisher and the consumer. For efficient caching distribution, several placement strategies were proposed. Thus, a number of diverse types of cache management strategies, such as max-gain in-network caching (MAGIC) [34], WAVE popularity-based caching strategy, hop-based probabilistic caching (HPC), LeafPopDown, most popular cache (MPC), and cache capacity aware caching (CCAC) are explained in the following sections.

Ren et al. developed MAGIC [34] to minimize the overall bandwidth consumption as well as the amount of caching operations. In MAGIC, content caching performs using two procedures. In the first procedure, a content is cached in the Content Store (CS) and in the second procedure, the content replacement operation is performed to delete the content from the router’s cache, so that the new content can be accommodated. The caching operation is based on the popularity of contents and the hop reduction count. In MAGIC, two primitives, local cache gain and max-gain values, are used to calculate the popularities of the requested contents. Each router locally calculates the local cache gain in which the numbers of placement and replacement operations are measured. MAGIC uses the local information (content name, interest count, popularity count) related to the contents to calculate the local gain. The local gain helps to find the appropriate node with location to cache the incoming contents along the data delivery path.

To find the router that has maximum local gain, MAGIC adds extra information, named max-local gain, and embeds it into the consumer interest. When the interest is generated for some content, the counter for max-gain value is initialized to 0. As the interest is received at any node, the MAGIC compares values of the max-gain of interest with local gain of the node. If the local gain value is greater than the value of max-gain, the node updates the max-gain value. As the interest along with its max-gain reaches the content provider, the max gain value is delivered to content and embedded into content. Therefore, the content is cached along the delivery path if the max-gain value of the transmitted content and the local gain value of the router are equal. Thus, the content with more interests has a greater chance to be cached at the intermediate node.
Wang et al. proposed HPC [36] to overcome the issues of content caching mechanisms. It was developed by using the enhanced version of CacheWeight_y factor and CacheWeight_{MRT}. The CacheWeight_y was enhanced to reduce the similar content replications and the y parameter was used to determine the number of hops between a content provider and a consumer. CacheWeight_y also helps to minimize the distance between the cached content and consumer by pushing the content towards the consumer to reduce the path length for subsequent interests [41].

\[
\text{CacheWeight}_y = \frac{1}{y + a}, \quad \alpha \geq 0
\]  
\[
\text{CacheWeight}_{MRT} = MRT_m + MRT_{exp}
\]  
\[
\text{HPC} = \text{CacheWeight}_y + \text{CacheWeight}_{MRT}
\]

The CacheWeight_y factor is equal to the distance between the consumer and the provider. Equation (1) shows the distance in hop-count and is used as constant integer that assigns to the cache capacity of the whole path. The CacheWeight_{MRT} refers to the duration that identifies for how long a content can be cached along the data delivery path. CacheWeight_{MRT} is derived using two factors as mean residence time (MRT_m) and expected mean residence time (MRT_{exp}). Suppose the value is selected as MRT_{exp} = 5 s, a = 1, and the value of parameter is associated to 1, 2, 3, and 4 with a path having nodes N1, N2, N3, and N4, respectively. The value of MRT_m factor is derived as 8, 3, 5, and 15 along the node N1, N2, N3, and N4, respectively. Therefore, CacheWeight_y factor is achieved as 0.5, 0.33, 0.15, and 0.2 and this value is dispersed along the data routing path with each node as N1 = 0.5, N2 = 0.33, N3 = 0.15, and N4 = 0.2, respectively.

Cho et al. designed WAVE [35] popularity-based caching strategy to achieve data distribution in chunks. The objective of the WAVE strategy is efficient distribution of contents as well as reducing the average cache management cost. In WAVE strategy, a content is cached in the form of chunks on the basis of the requirements of a consumer. All the chunks are associated with a relation known as inter-chunk relation. The WAVE strategy modifies the number of chunks imparted to a cache according to the popularity of the content (the number of interests received by a specific content). The WAVE strategy exponentially increases the number of chunks to be cached and gradually forwards the chunks towards the consumers. All the routers individually decide whether the received chunk has to be cached or not. In this strategy the upstream routers recommend to its downstream routers to cache the incoming chunk and this recommendation can be ignored by the router if the cache space of the downstream router is full. If the downstream router does not have cache space to store the coming chunk, the chunk will be stored at the subsequent downstream router. When the content provider receives an interest for a specific content or file, it divides the content into chunks to result in efficient data distribution. For instance, a requested content is divided into 50 chunks and the provider will send that content in the form of chunks to the consumer. All the chunks of a content have a specific type of index and the index is used in collaboration with the routers to avoid caching of disorganized chunks because each router individually makes a decision regarding the caching operation. Moreover, a cache flag bit (e.g., a data packet header) is associated with each chunk for organized data delivery.

Bernardini et al. developed MPC [42], in which each node is associated with a special entity known as a popularity table, and three types of information are needed to store the cached content: the content name, the popularity count, and the threshold. All nodes need to calculate the number of incoming interests for each content name to measure the popularity of content [43]. The threshold is the maximum value according to which the content is considered popular, and this value is fixed by the content caching mechanism. When the popularity count for a particular content name becomes equal to the threshold value, the content is labeled as popular. If a node holds the popular content, it recommends caching it at its neighbor nodes by sending a suggestion message. The suggestion message may or may not be acknowledged, depending on the resource’s (i.e., the cache’s) availability.
When popular content is cached at neighbor nodes, its popularity is reinitiated to avoid flooding caused by the replication of analogous content.

Lee et al. designed cache capacity aware caching strategy [39] by the composition of selective caching with the cache aware routing algorithm. It was designed to reduce the loads of networks and servers. CCAC observes the recent cache consumption to estimate the available cache to accommodate new coming contents to be cached at a routing path. A node indicating highest cache capacity has more chances to cache the upcoming content within its cache. The cache aware routing forwards the consumer’s interest to the suitable provider by using an extra feature as forwarding information base (FIB). The cache capacity is calculated using an inverse function to the amount of recently used cache. Each node indicates a heterogeneous size of cache capacity. Therefore, the cache capacity value of the $i$-th node can be calculated as shown in the following equation.

$$CCV(i) = \frac{c}{L(i)} \times \text{Cache}_{size}(i)$$

(4)

where $c$ represents the compensation value in the case that the huge caching load, $L$, makes a distortion among the network nodes. Two fields as $CCV_i$ and network distance value is integrated with each interest to find the available cache and distance between a consumer and a content provider. As the hit is recoded, the content is forwarded to the consumer according to following equation

$$w_r = \frac{\log r}{\log N_{total}} = \frac{\log N_{total}}{\log r}$$

(5)

where $N_{total}$ shows the total number of contents and $r$ represents the ranking of popular contents. The popularity of a content is measured by taking the sum of interests generated for a particular content. Therefore, a weight is associated to each popular content that helps the content to be cached with higher possibility. The $CCV_i$ is attached to the content as the provider makes a reply to the interest and the threshold is individually determined by combining the $CCV_i$ and $w_r$ at all nodes along the routing path.

$$CCV_{th} = CCV_{highest} \times w_r$$

(6)

where $CCV_{th}$ represents a threshold, which is used to make content popular.

Khattak et al. developed LeafPopDown [37] recently. It is a popularity-based cache management strategy used to manage the cache of nodes in NDN. LeafPopDown caches popular content at the downstream node and edge nodes among the data downloading path. When a consumer sends an interest for content, a path is created from the consumer to the provider node for further communication. As the cache hit occurs, the provider node adds an entry in its popularity table. As the entry for that particular content exceeds the threshold (value to identify the popular content) the content is suggested as popular. The popular content is sent back to the interested consumer, and a copy of the content is cached at a neighbor node, which is placed one level down from the provider node, and one copy is cached at an edge node located just before the interested consumer.

ProbCache is considered during the dynamic probabilistic caching in which the content caching decision is based on the consideration of cache capacity along the data routing path and the TimesIn factor [14]. The TimesIn factor is used to calculate the specific duration taken for the content to be cached. ProbCache increases the tendency of content caching near the consumer using the TimesIn factor and it increases the duration for a content to be cached at those routers, which are closer to the consumers [44]. Therefore, probabilistic caching decision is derived after the calculation of two factors (TimesIn and CacheWeight) as given below.

$$\text{TimesIn} = \frac{\sum_{n=1}^{x-y+1} N_n \ldots}{Ttw \times N_x}$$

(7)
Cache Weight = \frac{y}{x} \quad (8)

\text{ProbCache} = \text{TimesIn} + \text{CacheWeight} \quad (9)

where \( N_n \) represents the total cache capacity along the data routing path, \( N_x \) shows the average cache capacity of the total data routing path, \( T_{tw} \) indicates the target time window (for how long a content can be cached at a point), and \( n \) represents the number of routers along the path from the provider to the consumer. While calculating the CacheWeight factor, \( x \) and \( y \) represent the values of time since inception (TSI) and time since birth (TSB), respectively. Equation (1) calculates the duration for which a content can stay at a location for each router. Equation (2) measures the distance between the content provider and consumer and assigns different weights to the contents at all the routers. The weight increases with the decreasing distance from the consumers [45]. Consequently, the TimesIn factor demonstrates the time taken to transmit the cache content along the data routing path. Simultaneously, the CacheWeight factor is used to cache the content near the consumer.

3. The Problem Description

NDN caching is a revolution in network architecture. It can overcome the issues arising in the current Internet architecture. Moreover, it can reduce communication overhead, resource consumption, and bandwidth by caching popular content at multiple locations. However, it is difficult to decide which content needs to be cached at which location to produce effective and efficient results. Therefore, several cache management strategies [18,46–48] have been developed. Still, it is not clear which caching mechanism is the most ideal for each situation. MAGIC was established to reduce the bandwidth consumption and stretch (hop reduction). It also provides a solution for the problems of least recently used contents in different strategies. Caching contents locally need to be cached at the best possible position. It is concluded from the existing study that there is a basic need of a caching strategy that will place the transmitted contents at the optimal routers which will minimize the average cost, usage of the resources, and bandwidth consumption, and manage the cache more efficiently. MAGIC uses more cost and resources to compute max-gain and local gain values. Moreover, it consequently increases the content retrieval time and reduces the cache hit ratio. Moreover, it caches the popular content at limited locations that reduces the caching hit ratio because many interests need to send to the remote providers.

HPC was introduced to improve the content caching mechanism in terms of reducing the content redundancy. It associates specific time span to all the contents during their dissemination from a provider to a consumer. The caching duration is increased as the distance is decreased from the desired consumer [49,50]. Likewise, HPC also increases the homogeneous content replications at multiple locations. However, it does not mention any criteria to cache content when there is no free cache for the accommodation of new content. Therefore, the retrieval latency is increased because there is a possibility of caching contents far from the consumer that increases the stretch [51]. This strategy needs the additional computational cost and resource consumption because it requires extra parameters (TSI and TSB) to compute for all the contents at all nodes. Consequently, poor performance is generated for efficient content caching.

The WAVE caching strategy was developed to implement the chunk level caching to reduce the usage of resources and increase the amount of diverse contents to be cached at a location. However, it requires more time for caching entire chunks that belong to the same content and demonstrates a lesser cache hit ratio. Consequently, the WAVE strategy delivers moderate performance in terms of stretch ratio. Moreover, it does not define any distinction while caching the contents at a particular location. The WAVE strategy provides fast content dissemination; however, it distributes redundant and replicated content at multiple locations, exhibits higher resource consumption, no content distinction, and the content has to be continuously updated resulting in computational overhead.
MPC generates homogeneous content replication by caching the same content in multiple locations at all neighbors’ nodes. If the content provider and the consumer are associated with a small stretch path, then MPC increases its caching operations of redundant content due to the limited cache capacity. Hence, the hit ratio cannot satisfy the higher demand of consumers. However, if some content is popular again, then there is no distinction to stop the flooding of the same content in the network, which increases the usage of both the resource (cache) and the bandwidth. Moreover, it increases the usage of the cache storage to manage the popularity table for each content name at all the nodes, increasing the communication and searching overhead in order to decide which popular content will be cached at the neighbor nodes when different content has the same popularity within the limited cache size. Hence, the cache hit decreases.

In addition, MPC does consider the time needed to calculate content popularity. Moreover, no criterion is defined for choosing popular content according to time consumption. Let us assume that three interests for content C1 are generated in five seconds, and two interests for content C2 are sent in one second. According to MPC, content C1 will be most popular because the frequency of generating interests (e.g., the number of interests per second) is not considered when determining content popularity. Consequently, the most recently used content will remain unpopular, which increases the hit ratio due to network congestion. Furthermore, if the content is located far from the consumer and numerous interests are generated for that content, but its popularity is low, then the content will not be considered as popular and the interests will need to traverse several hops to find the required content, and they will face high bandwidth that increases the path length in terms of stretch.

CCAC claims to provide content caching services with content routing services. For both services, it uses several entities such as $CCV_i$, $CCV_{th}$, additional FIB, $w_r$ (content ranking), and popularity for contents. All these entities execute at all the nodes separately whenever an interest is generated, or data is transmitted to the consumers. In this way, the communication overhead is increased and consequently, the cache hit ratio keeps at its minimum level because millions of interests are generated and correspondingly contents are transmitted in a very short interval. Moreover, CCAC shows the content redundancy because it distributes cache capacity along the data routing path within all nodes and all the contents are replicated wherever they found cache space. Similar types of popular contents are cached at multiple nodes that decrease the overall diversity ratio.

The LeafPopDown content caching mechanism increases the amount of homogeneous content replication, which decreases the cache storage to accommodate the new content. It also increases the resource (cache) utilization by caching analogous content at different locations. Moreover, the LeafPopDown has not defined any criteria for caching less popular content, which increases the content redundancy ratio because a large amount of similar content is cached. In addition, it reduces content diversity because it repeats the caching of similar content throughout the data delivery path (provider to consumer). Most importantly, it decreases the cache hit ratio because it increases the amount of identical content within the network cache, which reduces the memory space available for diverse popular content. Therefore, most of the interests for popular contents need to be forwarded to the remote content provider.

The dynamic-based probabilistic caching strategies tries to provide fair resource allocations among the delivering contents; however, in all strategies, the interests and contents need to update their headers with TSI and TSB at each router, which increases the computational and communication overhead. The dynamic-based probabilistic caching strategies were developed to reduce content redundancy and path redundancy, however, these strategies failed to meet these goals [52]. All these strategies still exhibit a significant amount of redundancy. The ProbCache demonstrates low content diversity ratio because it caches all the requested contents at all the op-path nodes thus, there is a lesser chance to accommodate diverse content at multiple locations [53]. ProbCache caches all the transmitted content without considering the popularity of contents. Therefore, all the contents needed to be cached at all the nodes along the data delivery path. Consequently, when an interest is generated.
to download content it needs to traverse several hops to find the appropriate content from the distinct provider. Thus, the overall cache hit ratio is reduced and the stretch ratio is increased [49,50].

Regarding the critical issues with the existing caching mechanisms, the following questions needed to be identified:

- How can content be selected for caching regarding their popularity to increase the availability of diverse content to achieve a better cache hit ratio?
- How can the caching of diverse content be improved to reduce the amount of redundant content?
- How can the caching of required content availability be increased in terms to reduce the path stretch?

To answer these questions, a new caching mechanism is proposed in the next section.

4. Compound Popular Content Caching Strategy

NDN research approaches need to be verified and validated on a widely used platform (simulation environment). In previous studies [54–56] it was observed that the ideal structure of the network could affect the overall performance of the network. Cache management is an optimal feature of content centrism, and many researchers have focused on the diverse methods of managing disseminated content in networks. Recently, several content caching mechanisms were developed to increase the efficiency of in-network caching by managing the transmitted content according to the diverse nature of caching approaches. However, in existing caching mechanisms, several problems related to the multiple replications of heterogeneous content persist in increasing memory wastage. To actualize the basic concept of the NDN cache, content caching mechanisms must implement the optimal objectives and overcome the issues in the data dissemination process faced by the above caching mechanisms [57]. Consequently, in this study, a new flexible mechanism for content caching was designed to improve the overall caching performance [58]. This section provides a complete description of the proposed content caching mechanism, which is divided into two phases.

In the first phase, the optimal popular content (OPC) is selected by calculating the total number of received interests for a particular content name. All nodes calculate the number of received interests for each content using the Pending Interest Table (PIT) record and categorize the content into high-interested (the content that received most of the consumer interests) and least popular content (LPC) (the content that received least consumer interests) based on the threshold value. The threshold value is the average of the total number of received interests for all content, and it changes whenever a new interest is generated.

Therefore, the threshold value shows the maximum average of the total received interests for all content. If the number of received interests for a particular content name is greater than the average of the total number of received interests for all content, the content is considered to have a high level of interest. Let us assume that the sum of the received interests for all content is 30, in which 20 interests are generated for content C1, and 10 interests are received for content C2. Taking the average of all received interests, the threshold value is 15. In this case, content C1 is the OPC because it received more interests (20) than the threshold (15). Meanwhile, if the total number of received interests for a content item is less than the average of the total number of received interests for all content, then the content is considered as LPC. According to the content selection algorithm, all the LPC are sorted in ascending order and then one fourth of the total contents are selected as the most frequently interested recently. Few contents are selected to increase the cache hit ratio. Let us suppose that the total number of contents is 16 in the sorted content list, and then 4 contents will be selected as LPC. Algorithm 1 illustrates the selection of the OPC and LPC.
The second phase is related to the caching decision about the transmitted content. Information, such as the number of hops, the path distance (from the provider to the consumer), and the location (for caching the transmitted content), are required to cache the content at the intermediate nodes. In this proposed caching mechanism, the number of incoming and outgoing paths is calculated for each node to find the stretch mutual node whenever content is disseminated from the provider to the consumers. Let us assume that a consumer sends an interest to the network. However, the cache is empty at the nodes along the path toward the consumer. According to the general NDN practice, the interest needs to traverse until it matches the required data at the provider’s node. Therefore, the corresponding content is transported through the back path from the provider to the consumer. Furthermore, the copy of the OPC is cached at all the mutually connected nodes of the data routing path before the consumer to reduce the computational overhead and the high bandwidth cost for the dissemination of subsequent interests and the caching state of a mutual node is shared with its neighbor nodes via a broadcast message to inform them about the location of the OPC. However, the mutual positions of caching have promised better cache efficiency because most of the interests are satisfied there. Although no criteria were defined in earlier caching strategies to handle LPC, the proposed content caching mechanism defines a special criterion for caching to increase content dissemination efficiency and it increases the availability of diverse content close to the consumers. In the proposed mechanism, LPC is cached only at one mutual node that is placed near the consumers. To avoid the unnecessary usage of the cache, the LPC caches are only at one node because there is less of a chance to regenerate the interests for the LPC. Hence, most of the cache is used to accommodate the OPC. The Algorithm 2 illustrates the caching mechanism of OPC and LPC.

Algorithm 1 Selection of optimal popular content (OPC) and least popular content (LPC) content

```python
def GetPopularContent(OPC_List, LPC_List, interest_List):
    Icount = dict(interest, int);
    total_Contents = 0
    for content in interest_List:
        if Icount.Contains(content):
            Icount[content] += 1
        else:
            Icount.Add(content, 1)
            total_Contents += 1
    avg_RequestCount = len(interest_List)/total_Contents
    for content in Icount.keys(self):
        if Icount[content] > avg_RequestCount:
            OPC_List.Add(content)
        else:
            LPC_List.Add(content)
    LPC_List.Sort.ascend(LPC_List.values)
    LPC_List_To_Return = []
    content_Count_To_Return = (25/#total contents in LPC_List)*100
    i = 0
    for content in LPC_List:
        if i <= content_Count_To_Return:
            LPC_List_To_Return.Add(LPC_List[i])
        else:
            break;
        i += 1
    return OPC_List, LPC_List_To_Return
```

The second phase is related to the caching decision about the transmitted content. Information, such as the number of hops, the path distance (from the provider to the consumer), and the location (for caching the transmitted content), are required to cache the content at the intermediate nodes. In this proposed caching mechanism, the number of incoming and outgoing paths is calculated for each node to find the stretch mutual node whenever content is disseminated from the provider to the consumers. Let us assume that a consumer sends an interest to the network. However, the cache is empty at the nodes along the path toward the consumer. According to the general NDN practice, the interest needs to traverse until it matches the required data at the provider’s node. Therefore, the corresponding content is transported through the back path from the provider to the consumer. Furthermore, the copy of the OPC is cached at all the mutually connected nodes of the data routing path before the consumer to reduce the computational overhead and the high bandwidth cost for the dissemination of subsequent interests and the caching state of a mutual node is shared with its neighbor nodes via a broadcast message to inform them about the location of the OPC. However, the mutual positions of caching have promised better cache efficiency because most of the interests are satisfied there. Although no criteria were defined in earlier caching strategies to handle LPC, the proposed content caching mechanism defines a special criterion for caching to increase content dissemination efficiency and it increases the availability of diverse content close to the consumers. In the proposed mechanism, LPC is cached only at one mutual node that is placed near the consumers. To avoid the unnecessary usage of the cache, the LPC caches are only at one node because there is less of a chance to regenerate the interests for the LPC. Hence, most of the cache is used to accommodate the OPC. The Algorithm 2 illustrates the caching mechanism of OPC and LPC.
Algorithm 2 Caching of OPC and LPC Contents

1. Mutual(RequestedContentNode, RequestedContent, InterestNodes, allNodes, interestList)
2. RequestedContentNode
3. RequestedContent
4. InterestNodes=[], #Interest (shows consumer’s request)
5. allNodes=[]
6. for Node N in allNodes:
7.   for content c in N
8.     if c==RequestedContent Then
9.       RequestedContentNode=N
10.  pathToContentList=[]
11. for Node N in InterestNodes
12.   path=getPath(N, RequestedContentNode)
13.   pathToContentList.Add(path)
14. mutualNodes = GetMutualNodes(pathToContentList)
15. FindAndSetContentWithShortPath(mutualNodes)

Procedure: GetMutualNodes(pathToContentList)
16. #Get nodes with maximum users connected for all the paths
17. for node in pathToContentList.paths
18.   if node.ConnectedUsers.Count()>1
19.     mutualNodes.Add(node)
20. return mutualNodes

Procedure: FindAndSetContentWithShortPath(mutualNodes)
21. for Node N in mutualNodes
22.   for Node M in mutualNodes
23.     if N.Length<=M.Length Then
24.       n=N
25.       n.cache.contentPlacement(OPC)
26.       n.cache.contentPlacement(LPC)

To accommodate newly arriving content, CPCCS adopts the least recently used (LRU) content replacement policy to evict old content from the cache after its life span expires. For example, if a node has large cache storage, CPCCS will select five seconds as the content life span, meaning that, the content would be deleted after five seconds of cached time. However, if a node has less available storage, the content life span would decrease.

Figure 1 illustrates the basic content caching mechanism in CPCCS. In the given scenario, content provider P publishes content C1 and content C2 in a network. The content is initially cached at node N10. After a while, node N10 receives four interests from consumers A, B, and C. To respond to their interests, node N10 becomes a provider and sends content C1 to consumers A, B, and C. In Figure 1, the consumer’s interests are indicated by dotted line arrows. The solid line arrows indicate the responses from the provider. Simultaneously, two interests for content C2 are received from consumer D and consumer E at node N10. In response, N10 sends content C2 to consumer D and consumer E. According to the proposed caching mechanism, the total number of received interests in content C1 is six. Subsequently, content C1 is selected as OPC because it has an extra interest over the average of the total number of interests received for both content C1 and content C2. Therefore, the caching operations during the transmission of content C1 as OPC are done at mutually connected nodes N4 and N6 because both Node N4 and N6 are mutually connected with interested consumers as A, B, and C. Therefore, the subsequent interests from consumer A, B, and C will be satisfied from these mutually
connected nodes (N4 and N6). Secondly, the content C2 is selected as the LPC because it has fewer interests than the average of the total number of received interests for content C1 and content C2 at provider node N10. Therefore, content C2 will only be cached at one mutual node as N11, indicating the least distance from the interested consumers D and E as shown in Figure 1.

Through this way, the following goals are achieved: the content redundancy and content diversity are improved by increasing the caching of heterogeneous content, the stretch is kept at a minimum, and in turn the cache hit ratio is increased by caching the content near the consumer.

5. Performance Evaluation

NDN research approaches need to be verified and validated on a widely used platform (simulation environment). In previous studies [54–56] it was observed that the ideal structure of the network could affect the overall performance of the network. Cache management is an optimal feature of content centrism, and many researchers have focused on the diverse methods of managing disseminated content in networks. Recently, several content caching mechanisms were developed to increase the efficiency of in-network caching by managing the transmitted content according to the diverse nature of caching approaches. However, in existing caching mechanisms, several problems related to the multiple replications of heterogeneous content persist in increasing memory wastage.

To evaluate the proposed content caching mechanism CPCCS, it is necessary to compare its attributes with analogous ones in earlier mechanisms [18,42,47]. Therefore, we selected the SocialCCNSim-Master simulator [59] as the simulation environment for the performance comparison. SocialCCNSim-Master was specially designed to identify the performance of NDN caching. It takes data traffic from the Facebook social network topology, which is associated with 4039 users who each have 44 relationships, and 88,234 friends. SocialCCNSim-Master supports five internet service provider (ISP)-level topologies (i.e., Gigabit European Academic Network (GEANT)). GEANT topology was developed especially for research, education, and innovation communities across globe. It is a pan-European network used to interconnect Europe’s national and education networks (NRENs).
GEANT has the ability to connect more than 50 million consumers at 10,000 organizations. Moreover, GEANT provides a high bandwidth and secure high capacity (50,000 km) network with an increasing range of services that allow researchers to cooperate with one another wherever they are located. GEANT has 22 nodes connected to each other. The objective of the GEANT topology is to transfer large amounts of data across the nodes. Its structure is perfect for testing and comparing our simulation parameters, recording the cache hits, and observing the efficacy of the CPCCS.

To avoid unnecessary cache usage, the least replacement policy (LRU) was used to replace the old content with the new. The LRU is considered the most efficient content replacement policy because of its flexible structure and high performance. In this study, four different content categories were selected based on their popularity level, as defined through Zipf’s distribution: User generated content (UGC) and VoD [60] were used because they showed high traffic production due to consumer interest in media-driven content. The Zipf content probability distribution function was used to select the content category. Due to the cache-driven architecture of NDN, various algorithms and strategies have been designed to choose from based on the set of rules regarding which content to cache from the varieties of popular content, content-defined names, and topological content information, among others. For the purposes of this study, the research will be caching sets of content of two categories, user-generated content (UGC) and video on demand (VoD), based on their popularity defined through Zipf’s distribution. The present study used Zipf 0.8 and 1.2 for UGC and VoD content, respectively, because it shows high traffic production due to consumer interest in media-driven content [61].

The cache size (i.e., the amount of space used to store content temporarily during its transmission) ranged from 1 GB to 10 GB, and the catalog size was $10^8$ elements of 10 MB each. The simulation scenario was divided into 10 equal steps, as shown on the x-axis in all of the simulation graphs. Each step concluded after 24 hours of simulation. The graphical representation of the consequences was achieved by taking the average result of all simulations.

### 5.1. Cache Hit Ratio

The cache hit ratio is the key metric in evaluating the performance of the NDN cache. It refers to the response by the in-network cache storage in which the content is locally cached for a specific time period [62]. The cache hit occurs when a consumer’s required content is found within the network node’s cache. Therefore, the cache reacts as a provider, sending the corresponding content to the appropriate consumer [63]. It can be defined as follows:

$$\text{Cache Hit Ratio} = \frac{\sum_{n=1}^{N} \text{hit}_i}{\sum_{n=1}^{N} (\text{hit}_i + \text{miss}_i)}$$

(10)

Figures 2 and 3 show the results created by using the SocialCCNSim simulator for the cache hit ratio, demonstrating that the CPCCS performed better than the benchmark caching mechanisms. CPCCS showed good quality throughout the simulation results in all content categories (i.e., VoD, UGC). This is because CPCCS caches OPC close to the consumers, which increases the availability of the most desired content for subsequent responses. Another benefit of CPCCS is that it initializes a time span for each content, which indicates how long the content can be cached at a specific location (i.e., node). Therefore, CPCCS decreases the unnecessary usage of cache storage and increases its ability to accommodate new content by removing unnecessary content. CPCCS also improves the caching of heterogeneous content by selecting the OPC and LPC.
WAVE, MAGIC, LeafPopDown, and HPC show similar performance in terms of cache hit ratio with small and large cache size. However, HPC shows better cache hit ratio due to its selection of caching content on all path nodes for a specific time span. Meanwhile, MPC performed better than WAVE and CACC, but its results were not as favorable as those of CPCCS because MPC took longer to select popular content. MPC uses a popularity table for each content, which increases the searching overhead when calculating which content is the most popular. Moreover, MPC cached the most content.
searching overhead when calculating which content is the most popular. Moreover, MPC cached the most popular content at all neighbor nodes, which increased the unnecessary usage of cache storage and lessened its chances of accommodating new incoming content; thus, the overall hit ratio was decreased. CACC performed slightly worse than MPC because it cached all of the content regardless of the consumer interest level at the intermediate node, increasing the chance of accommodating the least popular content.

WAVE depicts lower cache hit ratio because it increases the amount of similar contents along the data downloading path and it consumes extra time to bring the required content. Therefore, interests for diverse content need to be forwarded to the main provider, which makes for a long trip in content downloading. Moreover, the ProbCache shows poor ratio of cache hit because it does not consider the popularity of content for caching operations. Therefore, all the contents need to cache at all the nodes along the data delivery path. Whenever, the new interest is generated, it needs to be forwarded to the remote provider to accomplish its requirements. However, CPCCS shows better performance in terms of cache hit ratio because it provides the most frequently interested contents from the desired and near consumer. When we expand the cache size, CPCCS again shows better results as the cache hit ratio increases because of its nature in caching heterogeneous content along the routing path. From the numerical results shown in Table 1, we may conclude that CPCCS performs better in terms of cache hit ratio than the comparing strategies.

### Table 1. Numerical values on cache hit ratio.

| Category                | User Generated Content (UGC) ($\alpha = 0.88$) | VoD Content ($\alpha = 1.2$) |
|------------------------|-----------------------------------------------|-----------------------------|
| Strategies             |                                               |                             |
| ProbCache              | 0.32 0.33 0.34                                | 0.29 0.30 0.31              |
| WAVE                   | 0.34 0.35 0.36                                | 0.30 0.31 0.32              |
| CACC                   | 0.36 0.38 0.40                                | 0.32 0.33 0.34              |
| MPC                    | 0.35 0.39 0.44                                | 0.32 0.35 0.37              |
| MAGIC                  | 0.36 0.37 0.38                                | 0.33 0.34 0.35              |
| LeafPopDown            | 0.37 0.39 0.41                                | 0.33 0.35 0.37              |
| HPC                    | 0.38 0.42 0.46                                | 0.34 0.36 0.38              |
| CPCCS                  | 0.39 0.48 0.56                                | 0.35 0.39 0.44              |

5.2. Content Diversity

Content diversity refers to the amount of heterogeneous content that is accommodated at certain locations. It also refers to the different types of content residing within the cache along the data delivery path [64]. Content diversity can be calculated as follows:

$$Diversity = \frac{\sum n(n-1)}{N(N-1)}$$  \hspace{1cm} (11)

where $n$ is the amount of content of a particular type, and $N$ is the total amount of content in the network. The ratio of the diverse content accumulates in a network cache or different content can be stored in a cache to increase content diversity [65].

The replication of analogous content increases content redundancy, which causes network congestion and high bandwidth. In contrast, diversity reduces the amount of similar content replication. Diversity provides the ratio between the amount of unique and similar content that is accommodated at the same cache location. One of the basic objectives of CPCCS is to minimize the high number of analogous content replications to increase the amount of diverse content. Figures 4 and 5 illustrate the results of content diversity in which HPC shows the lowermost diversity ratio, which means that the replication of homogeneous contents in HPC is higher than other strategies. The reason is that the HPC caches all the contents at all on-path nodes. Moreover, when we enlarge the cache size from 1 GB to 10 GB, HPC achieves a slightly better diversity ratio with both content categories.
(UGC and VoD). The reason is that, it does not reduce the replications done by similar contents. In the same way, it shows less quantity of diversity than all comparing strategies as shown by numeric values in Table 2. CCAC shows a higher diversity ratio than HPC, because it reduces the homogeneous content replications more than HPC.

**Table 2.** Numerical values on content diversity.

| Content       | UGC (α = 1.0) | VoD (α = 1.2) |
|---------------|---------------|---------------|
| Cache-Size    | 1 GB 5 GB 10 GB | 1 GB 5 GB 10 GB |
| ProbCache     | 0.37 0.39 0.41 | 0.38 0.40 0.42 |
| HPC           | 0.40 0.42 0.44 | 0.37 0.38 0.39 |
| CACC          | 0.43 0.45 0.47 | 0.41 0.43 0.46 |
| MPC           | 0.46 0.50 0.55 | 0.46 0.49 0.51 |
| WAVE          | 0.44 0.46 0.48 | 0.47 0.49 0.52 |
| LeafPopDown   | 0.46 0.47 0.48 | 0.47 0.48 0.49 |
| MAGIC         | 0.47 0.52 0.57 | 0.47 0.51 0.57 |
| CPCCS         | 0.52 0.60 0.68 | 0.57 0.65 0.74 |

However, it shows little diversity due to the higher number of similar replications of content than MPC. As compared to other strategies, MPC performs in a different way to some extent, because of its popularity method of caching content. It performs better with a large cache size than other strategies because it caches content only at neighbor nodes and keeps the other nodes empty along the data routing path. As a result, with a large cache size, supplementary content can be accommodated with diverse nature. WAVE performs better than CACC, HPC, and ProbCache because these strategies take a long duration to distribute similar content at all on-path nodes. In ProbCache the cache storage is used by unpopular contents and each content is replicated at all the nodes along the data routing path that reduces the ratio of diverse contents.

![Figure 4. Content diversity with user generated content (UGC).](image-url)
The WAVE increases the amount of diverse content because its caching decision is performed in chunks and hence less storage is used by chunks that increase the free cache to accommodate more content. LeafPopDown showed smaller diversity because the replication of the analogous content was much higher than in the other mechanisms. However, MAGIC shows better results in achieving a higher demand of content diversity. The reason is that, MAGIC caches a copy of transmitted content at limited locations. Therefore, the amount of free cache increases to accommodate the diverse content.

Compared to other strategies, CPCCS performs better in attaining a higher diversity ratio. The goal of CPCCS is to increase the amount of diverse contents by selecting the content based on their diverse popularity. It is concluded from Table 2 that CPCCS achieved a higher diversity ratio of contents than all other strategies compared.

5.3. Stretch

The distance traveled by a consumer interest toward the content provider is known as stretch [66,67]. The stretch can be calculated by the following equation:

\[
\text{Stretch} = \frac{\sum_{i=1}^{I} \text{Hop} - \text{traveled}}{\sum_{i=1}^{I} \text{Total} - \text{Hop}}
\]  

where \(\sum_{i=1}^{I} \text{Hop} - \text{traveled}\) shows the number of hops covered by a consumer interest between the consumer and the content-provider node; \(\sum_{i=1}^{I} \text{Total} - \text{Hop}\) represents the total number of hops between the consumer and provider and \(I\) illustrates the total number of generated interests for a specific content.

CPCCS caches the popular content close to the consumers at a central position (stretch centrality node), from where all the desired consumers can get their required content. Therefore, it makes the distance between a consumer and a provider smaller and most of the consumer interests travel through the central position and are satisfied from central positions. Moreover, CPCCS selects diverse popular content to be cached close to the consumers that increases the overall stretch ratio. On the other hand, all the comparing strategies show a long stretch path as shown in Figures 6 and 7.
ProbCache showed slightly better results than WAVE in terms of stretch because it caches all the contents at all nodes and therefore, the subsequent interests are satisfied from nearer nodes. Hence, the overall stretch is reduced. WAVE shows the larger path stretch with all cache sizes because it caches the popular content on the downstream node first, which increases the length between the consumer and provider. WAVE brings content close to the consumer, but it takes a long time because of its nature of caching decision.
As compared to MAGIC and CACC, HPC performs better with small and large cache sizes because it caches a copy of required content at all on-path nodes. Therefore, the subsequent interests are satisfied from the nearer cached copy of required content. However, MPC and LeafPopDown produce better results in terms of reducing the stretch because these strategies cache the required contents close to the consumers. From the results shown in Table 3, it is clear that the proposed caching strategy performs better than the all other strategies. CPCCS achieved enhanced results with all cache sizes (1 GB to 10 GB).

Table 3. Numerical values on stretch.

| Strategies     | UGC ($\alpha = 0.88$) | VoD ($\alpha = 1.2$) |
|----------------|------------------------|-----------------------|
| ProbCache      | 0.61 0.59 0.56         | 0.54 0.52 0.50        |
| WAVE           | 0.60 0.59 0.58         | 0.56 0.55 0.54        |
| MAGIC          | 0.57 0.55 0.53         | 0.53 0.48 0.44        |
| CACC           | 0.56 0.54 0.52         | 0.53 0.47 0.43        |
| MPC            | 0.54 0.52 0.50         | 0.49 0.44 0.40        |
| HPC            | 0.53 0.40 0.47         | 0.48 0.43 0.39        |
| LeafPopDown    | 0.53 0.49 0.44         | 0.47 0.42 0.38        |
| CPCCS          | 0.48 0.29 0.30         | 0.39 0.34 0.29        |

6. Conclusions

The exponential increase in Internet traffic can be controlled by the implementation of cache storage within the network nodes, allowing content to be cached for a certain time span inside these nodes to fulfill the requirements of subsequent interests. As such, caching has the ability to improve the overall performance of the network architecture. Based on the results of the comparative performance analysis, the current study proposes a new mechanism for managing the transformation of the content in a network. The purpose of this study was to implement a new cache management mechanism to enhance the content caching in NDN. Therefore, CPCCS was proposed to resolve the issues of previous caching mechanisms, such as MAGIC, MPC, LeafPopDown, CACC, HPC, WAVE, and ProbCache. The proposed mechanism provides for the compound caching of both OPC and LPC. To evaluate the proposed mechanism, a simulation environment was created using an NDN caching-based simulator, SocialCCNSim-Master. In this simulator, the caching performance in terms of cache hit ratio, content diversity, and stretch ratio were evaluated. Results show that CPCCS performed much better than the other caching mechanisms.

7. Future Directions

The NDN caching delivers benefits to Internet technologies through the implementation of cache storage. To handle the cache storage, it is most significant to improve the overall caching performance by deploying the efficient cache deployment strategy. Therefore, the proposed caching strategy enhances the caching performance by improving the content selection and content caching mechanisms. According to future prospective, this caching strategy could improve the content dissemination in several technologies such as Internet of things (IoT) [18], edge cloud computing [19], blockchain [20], distributed fog computing [21], software defined networking (SDN) [22,68], and fifth generation (5G) mobile cellular networks [23]. In regard to genomic data, the extensive use of genomic data sets that are immense, and rapidly increasing over time, are creating problems regarding their transmission. The reason is that these types of data sets are stored at distant (remote) databases and their transmission is supported by IP-based Internet. Therefore, similar genomic data sets are needed to be sent across the remote location several times [69,70]. Consequently, several problems are created regarding transmission of genomic data sets in which bandwidth, congestion, delay, and high resource consumption is maximized. In this situation, NDN provides a number of caching strategies to handle the issues related to genomic data transmission [28]. Therefore, the present caching strategy can solve...
these issues in an efficient way by caching the most popular and least popular content near the desired locations. Moreover, it can increase the availability of the most popular genomic data sets to respond locally to the subsequent requests. Hence, the congestion reduces, and bandwidth is used in an efficient manner. In addition, the resource (memory and energy) consumption will be decreased.

Author Contributions: Conceptualization, M.A.N.; data curation, M.A.N.; formal analysis, M.A.N.; funding acquisition, B.-S.K.; methodology, M.A.N., S.A.N. and S.H.; resources, B.-S.K.; supervision, B.-S.K., S.A.N. and S.H.; validation, M.A.N.; writing—original draft, M.A.N.; writing—review and editing, B.-S.K., S.A.N. and S.H.

Funding: This research was supported by the National Research Foundation of Korea (NRF) through the Korea Government (No: 2018R1A2B6002399).

Acknowledgments: This research was supported by the National Research Foundation of Korea (NRF) through the Korea Government (No: 2018R1A2B6002399).

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

References
1. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. IEEE Commun. Mag. 2012, 50, 26–36. [CrossRef]
2. Morley, J.; Widdicks, K.; Hazas, M. Digitalisation, energy and data demand: The impact of Internet traffic on overall and peak electricity consumption. Energy Res. Soc. Sci. 2018, 38, 128–137. [CrossRef]
3. Jat, D.S.; Bishnoi, L.C.; Nambahu, S. An Intelligent Wireless QoS Technology for Big Data Video Delivery in WLAN. Int. J. Ambient Comput. Intell. 2018, 9, 1–14. [CrossRef]
4. Liu, W.; Zhang, G.; Gao, Q. Coupling DAS, SVC and NDN: An SVC-aware cache and forwarding policy for NDN routers. In Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), Shenzhen, China, 15–17 August 2018.
5. Eastham, G.; Wong, O.; Chen, A. Identifying and Assigning Metrics to Influential User Generated Content. Google Patents US 9, 762, 428 B2, 12 September 2017.
6. Fan, Q.; Yin, H.; Min, G.; Yang, P.; Luo, Y.; Lyu, Y.; Huang, H.; Jiao, L. Video delivery networks: Challenges, solutions and future directions. Comput. Electr. Eng. 2018, 66, 332–341. [CrossRef]
7. Zami, T.; Morea, A.; Pasic, J. Benefit of progressive deployment of regenerators along with traffic growth in WDM elastic networks. In Proceedings of the Optical Fiber Communication Conference, San Diego, CA, USA, 11–15 March 2018.
8. Muhammad Ali Naeem, S.A.N.; Suhaidi, H. A Survey on Probabilistic Caching Mechanisms in Content Centric Networking. J. Adv. Res. Dyn. Control Syst. 2018, 10, 1309–1321.
9. Cui, X.-D.; Huang, T.; Jiang, L.I.U.; Li, L.I.; Chen, J.Y.; Liu, Y.J. Design of in-network caching scheme in CCN based on grey relational analysis. J. China Univ. Posts Telecommun. 2014, 21, 1–8. [CrossRef]
10. Hoogesteger, M.; Schmidt, R.D.O.; Pras, A. ITSA: Internet traffic statistics archive. In Proceedings of the Network Operations and Management Symposium (NOMS), Istanbul, Turkey, 25–29 April 2016.
11. Terpstra, R.D. Registration of Sip-Based Communications in a Hosted Voip Network. Google Patents US 2018/0287849 A1, 4 October 2018.
12. Vasilakos, A.V.; Li, Z.; Simon, G.; You, W. Information centric network: Research challenges and opportunities. J. Netw. Comput. Appl. 2015, 52, 1–10. [CrossRef]
13. Yi, C.; Afanasyev, A.; Wang, L.; Zhang, B.; Zhang, L. Adaptive forwarding in named data networking. In ACM SIGCOMM Computer Communication Review; ACM: New York, NY, USA, 2012; Volume 42, pp. 62–67.
14. Zhu, Y.; Mi, Z.-K.; Wang, W.-N. A Probability Caching Decision Policy with Evicted Copy up in Content Centric Networking. J. Internet Technol. 2015, 99, 1–11.
15. Bari, M.F.; Chowdhury, S.R.; Ahmed, R.; Boutaba, R.; Mathieu, B. A survey of naming and routing in information-centric networks. IEEE Commun. Mag. 2012, 50, 44–53. [CrossRef]
16. Amadeo, M.; Campolo, C.; Quevedo, J.; Corujo, D.; Molinaro, A.; Iera, A.; Aguiar, R.L.; Vasilakos, A.V. Information-centric networking for the internet of things: Challenges and opportunities. IEEE Netw. 2016, 30, 92–100. [CrossRef]
17. Yuan, H.; Crowley, P. Experimental evaluation of content distribution with NDN and HTTP. In Proceedings of the 2013 Proceedings IEEE INFOCOM, Turin, Italy, 14–19 April 2013.
18. Naeem, M.; Ali, R.; Kim, B.S.; Nor, S.; Hassan, S. A periodic caching strategy solution for the smart city in information-centric Internet of Things. *Sustainability* 2018, 10, 2576. [CrossRef]

19. Amadeo, M.; Campolo, C.; Molinaro, A. NDNe: Enhancing named data networking to support cloudification at the edge. *IEEE Commun. Lett.* 2016, 20, 2264–2267. [CrossRef]

20. Jin, T.; Zhang, X.; Liu, Y.; Lei, K. BlockNDN: A bitcoin blockchain decentralized system over named data networking. In Proceedings of the 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 4–7 July 2017.

21. Song, F.; Ai, Z.Y.; Li, J.J.; Pau, G.; Collotta, M.; You, I.; Zhang, H.K. Smart collaborative caching for information-centric IoT in fog computing. *Sensors* 2017, 17, 2512. [CrossRef] [PubMed]

22. van Adrichem, N.L.; Kuipers, F.A. NDNFlow: Software-defined named data networking. In Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), London, UK, 13–17 April 2015.

23. Jin, T.; Zhang, X.; Liu, Y.; Lei, K. BlockNDN: A bitcoin blockchain decentralized system over named data networking. *Sensors* 2017, 17, 2512. [CrossRef] [PubMed]

24. Song, F.; Ai, Z.Y.; Li, J.J.; Pau, G.; Collotta, M.; You, I.; Zhang, H.K. Smart collaborative caching for information-centric IoT in fog computing. *Sensors* 2017, 17, 2512. [CrossRef] [PubMed]

25. Qin, Y.; Yang, W.; Liu, W. A Probability-based Caching Strategy with Consistent Hash in Named Data Networking. In Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), Shenzhen, China, 15–17 August 2018.

26. 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]

27. Qazi, F.; Khalid, O.; Rais, R.N.B.; Khan, I.A. Optimal Content Caching in Content-Centric Networks. *Wirel. Commun. Mob. Comput.* 2019, 2019. [CrossRef]

28. Shannigrahi, S.; Fan, C.; Papadopoulos, C.; Feltus, A. NDN-SCI for managing large scale genomics data. In Proceedings of the ICN’18, Boston, MA, USA, 21–23 September 2018.

29. Yan, Q.; Yu, F.R.; Gong, Q.; Li, J. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Commun. Surv. Tutor.* 2016, 18, 602–622. [CrossRef]

30. Huo, R.; Yu, F.R.; Huang, T.; Liu, J.; Liu, Y. Software defined networking, caching, and computing for green wireless networks. *IEEE Commun. Mag.* 2016, 54, 185–193. [CrossRef]

31. Liu, W.-X.; Zhang, J.; Liang, Z.W.; Peng, L.X.; Cai, J. Content popularity prediction and caching for ICN: A deep learning approach with SDN. *IEEE Access* 2018, 6, 5075–5089. [CrossRef]

32. Jmal, R.; Fourati, L.C. An OpenFlow architecture for managing content-centric-network (OFAM-CCN) based on popularity caching strategy. *Comput. Stand. Interfaces* 2016, 51, 22–29. [CrossRef]

33. Ravindran, R.; Liu, X.; Chakraborti, A.; Zhang, X.; Wang, G. Towards software defined icn based edge-cloud services. In Proceedings of the 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet), San Francisco, CA, USA, 11–13 November 2013.

34. Ren, J.; Qi, W.; Westphal, C.; Wang, J.; Lu, K.; Liu, S.; Wang, S. Magic: A distributed max-gain in-network caching strategy in information-centric networks. In Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 27 April–2 May 2014.

35. 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 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Orlando, FL, USA, 25–30 March 2012.

36. Wang, Y.; Xu, M.; Feng, Z. Hop-based probabilistic caching for information-centric networks. In Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, USA, 9–13 December 2013.

37. Khattak, H.; UI Amin, N.; ud Din, I.; Insafullah; Iqbal, J. LeafPopDown: Leaf Popular Down Caching Strategy for Information-Centric Networking. *Int. J. Adv. Comput. Sci. Appl.* 2018, 9, 148–151. [CrossRef]

38. Bernardini, C.; Silverston, T.; Festor, O. MPC: Popularity-based caching strategy for content centric networks. In Proceedings of the 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013.

39. Lee, S.-W.; Kim, D.; Ko, Y.B.; Kim, J.H.; Jang, M.W. Cache capacity-aware CCN: Selective caching and cache-aware routing. In Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, USA, 9–13 December 2013.
40. Islam, S.; Moon, A.R. Analysis of LCD (Leave Copy Down) & LCE (Leave Copy Everywhere) Caching Scheme for Tree Topology. Ph.D. Thesis, East West University, Dhaka, Bangladesh, 2016.

41. Wu, H.; Li, J.; Zhi, J. MBP: A max-benefit probability-based caching strategy in information-centric networking. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015.

42. Bernardini, C. Stratégies de Cache Basées Sur la Popularité Pour Content Centric Networking. Ph.D. Thesis, Université de Lorraine, Nancy/Metz, France, 2015.

43. Dabirmoghaddam, A.; Barjough, M.M.; Garcia-Luna-Aceves, J.J. Understanding optimal caching and opportunistic caching at the edge of information-centric networks. In Proceedings of the 1st ACM Conference on Information-Centric Networking, Paris, France, 24–26 September 2014; pp. 47–56.

44. Namba, M. A recursive estimation of network state for improving probabilistic caching. In Proceedings of the 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR), Fukuoka, Japan, 13–15 November 2015.

45. Ascigil, O.; Sourlas, V.; Psaras, I.; Pavlou, G. A native content discovery mechanism for the information-centric networks. In Proceedings of the 4th ACM Conference on Information-Centric Networking, Berlin, Germany, 26–28 September 2017.

46. Naeem, M.A.; Nor, S.A.; Suhaidi, H. Information Centric Progressive Content Placement Cache for Multimedia data contents. *J. Adv. Res. Dyn. Control Sys.* 2018, 10, 1291–1301.

47. Naeem, M.A.; Nor, S.A.; Hassan, S.; Kim, B.S. Performances of Probabilistic Caching Strategies in Content Centric Networking. *IEEE Access* 2018, 6, 58807–58825. [CrossRef]

48. Naeem, M.A.; Nor, S.A. A survey of content placement strategies for content-centric networking. In *AIP Conference Proceedings*; AIP Publishing: Melville, NY, USA, 2016.

49. Chen, Q.; Shigeyasu, T. Transport control strategies in named data networking: A survey. *IEEE Commun. Surv. Tutor.* 2016, 18, 2052–2083. [CrossRef]

50. Zhang, M.; Luo, H.; Zhang, H. A survey of caching mechanisms in information-centric networking. *IEEE Commun. Surv. Tutor.* 2015, 17, 1473–1499. [CrossRef]

51. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsiropoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* 2014, 16, 1024–1049. [CrossRef]

52. Arianfar, S.; Nikander, P.; Ott, J. On content-centric router design and implications. In Proceedings of the Re-Architecting the Internet Workshop, Philadelphia, PA, USA, 30 November 2010.

53. Li, Y.; Zhang, T.; Xu, X.; Zeng, Z.; Liu, Y. Content popularity and node level matched based probability caching for content centric networks. In Proceedings of the 2016 IEEE/CIC International Conference on Communications in China (ICCC), Chengdu, China, 27–29 July 2016.

54. Carofiglio, G.; Mekinda, L.; Muscariello, L. LAC: Introducing latency-aware caching in Information-Centric Networks. In Proceedings of the 2015 IEEE 40th Conference on Local Computer Networks (LCN), Clearwater Beach, FL, USA, 26–29 October 2015.

55. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache “less for more” in information-centric networks. In Proceedings of the International Conference on Research in Networking, Prague, Czech Republic, 21–25 May 2012.

56. Chang, C.Y.; Chang, M.S. A hybrid coordination approach of in-network caching for named data networking. *Int. J. Future Gener. Commun. Netw.* 2016, 9, 285–300. [CrossRef]

57. Dräxler, M.; Karl, H. Efficiency of on-path and off-path caching strategies in information centric networks. In Proceedings of the 2012 IEEE International Conference on Green Computing and Communications (GreenCom), Besancon, France, 20–23 November 2012.

58. Zhang, G.; Li, Y.; Lin, T. Caching in information centric networking: A survey. *Comput. Netw.* 2013, 57, 3128–3141. [CrossRef]

59. Hamdi, M.M.F.; Habbal, A.; Zakaria, N.H.; Hassan, S. Evaluation of Caching Strategies in Content-Centric Networking (CCN) for Mobile and Social Networking Environment. *J. Telecommun. Electron. Comput. Eng.* 2018, 10, 1–6.

60. Bernardini, C.; Silverston, T.; Festor, O. A comparison of caching strategies for content centric networking. In Proceedings of the 2015 IEEE Conference on Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015.
61. Piro, G.; Ciancaglini, V.; Loti, R.; Grieco, L.A.; Liquori, L. Providing Crowd-Sourced and Real-Time Media Services through an NDN-Based Platform. In *Modeling and Processing for Next-Generation Big-Data Technologies*; Springer: Cham, Switzerland, 2015; pp. 405–441.

62. Chen, Z.; Pappas, N.; Kountouris, M. Probabilistic caching in wireless D2D networks: Cache hit optimal versus throughput optimal. *IEEE Commun. Lett.* **2017**, *21*, 584–587. [CrossRef]

63. Saino, L.; Psaras, I.; Pavlou, G. Icarus: A caching simulator for information centric networking (ICN). In Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques, Lisbon, Portugal, 17–19 March 2014.

64. Nguyen, Q.N.; Arifuzzaman, M.; Yu, K.; Sato, T. A Context-Aware Green Information-Centric Networking Model for Future Wireless Communications. *IEEE Access* **2018**, *6*, 22804–22816. [CrossRef]

65. Banerjee, A.; Banerjee, B.; Seetharam, A.; Tellambura, C. Content search and routing under custodian unavailability in information-centric networks. *Comput. Netw.* **2018**, *141*, 92–101. [CrossRef]

66. Badenhop, C.W.; Graham, S.R.; Ramsey, B.W.; Mullins, B.E.; Mailloux, L.O. The Z-Wave routing protocol and its security implications. *Comput. Secur.* **2017**, *68*, 112–129. [CrossRef]

67. Wang, S.; Bi, J.; Wu, J. Collaborative caching based on hash-routing for information-centric networking. In *ACM SIGCOMM Computer Communication Review*; ACM: New York, NY, USA, 2013; Volume 43, pp. 535–536.

68. Mahmood, A.; Casetti, C.; Chiasserini, C.F.; Giaccone, P.; Härr, J. Efficient caching through stateful SDN in named data networking. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, e3271. [CrossRef]

69. Reali, G.; Femminella, M.; Nunzi, E.; Valocchi, D. Genomics as a service: A joint computing and networking perspective. *Comput. Netw.* **2018**, *145*, 27–51. [CrossRef]

70. Femminella, M.; Reali, G.; Valocchi, D. Genome Centric Networking: A network function virtualization solution for genomic applications. In Proceedings of the 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, Italy, 3–7 July 2017.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).