Article

Path Segmentation-Based Hybrid Caching in Information-Centric Networks

Wei Li ^{1,2}, Peng Sun ^{1,2} and Rui Han ^{1,2,*}

¹ National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China; liw@dsp.ac.cn (W.L.); sunp@dsp.ac.cn (P.S.)

² School of Electronic, Electrical and Communication Engineering, University of Chinese Academy Sciences, Beijing 100049, China

* Correspondence: hanr@dsp.ac.cn; Tel.: +86-1-381-107-7380

Abstract: Information-centric networks (ICNs) have received wide interest from researchers, and in-network caching is an important characteristic of ICN. The management and placement of contents are essential due to cache nodes' limited cache space and the huge Internet traffic. This paper focuses on coordinating two cache metrics, namely user access latency and network resource utilization, and proposes a hybrid caching scheme called the path segmentation-based hybrid caching scheme (PSBC). We temporarily divide each data transmit path into a user-edge area and non-edge area. The user-edge area adopts a heuristic caching scheme to reduce user access latency. In contrast, the non-edge area implements caching network content migration and optimization to improve network resource utilization. The simulation results show that the proposed method positively affects both the cache hit ratio and access latency.

Keywords: ICN; hybrid caching; cache placement; content migration

Citation: Li, W.; Sun, P.; Han, R. Path Segmentation-Based Hybrid Caching in Information-Centric Networks. *Future Internet* **2021**, *13*, 16. <https://doi.org/10.3390/fi13010016>

Received: 17 December 2020

Accepted: 11 January 2021

Published: 12 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 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 (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Information-centric networks (ICNs) is the main direction of future network research. Current Internet architecture depends on the end-to-end correspondence between hosts; this is referred to as host-centric networks [1]. At present, the number of Internet users and traffic from multimedia applications are overgrowing, and users' primary demand for the web has changed to access massive content information. Host-centric networks do not adequately address the current needs, such as content distribution, mobility, etc. The features of the ICN are an excellent solution to this transition. ICN architectures utilize content naming, in-network caching, and a request-response model to achieve an effective and dependable content distribution [2].

Several ICN architectures have been proposed, such as DONA [3], NDN [4], PURSUIT [5], and NetInf [6]. In-network caching is an essential part of every ICN. An ICN node caches content by configuring additional storage. The user's requests can be responded to by the network node that caches the related content, thereby significantly reducing bandwidth utilization and content access latency. To improve an ICN caching system's overall efficiency, from the user's perspective, popular content should be copied to the edge of the network. From the overall network perspective, the cache's diversity should be increased as much as possible [7]. An excellent caching scheme requires effective coordination between these two targets [8].

Caching schemes in ICN can be divided into several categories, including cooperative and non-cooperative caching, off-path, and on-path caching [9]. In cooperative caching, cache nodes make cooperative caching decisions based on network topology, as well as partial or global cache status [8], while in non-cooperative caching, each cache node makes caching decisions individually. Compared to non-cooperative caching, cooperative caching requires additional communication and computation overhead but is more practical. In

on-path caching, content is cached on nodes on the data return path [10], while off-path caching [11] provides additional caching at a strategic location. On-path caching is more flexible and scalable than off-path caching.

This paper focuses on combining the advantages of on-path caching and cooperative caching and proposes a caching scheme based on path segmentation referred to as the path segmentation-based hybrid caching scheme (PSBC). We choose content migration between nodes to implement cooperative caching. Our goal is to coordinate the two cache targets described above with good performance in user access latency and network resource utilization. The main contributions of this paper are as follows:

- We propose a caching idea of path segmentation. We divide the cache nodes on the data return path into a user-edge area and a non-edge area according to the data transmission distance. Different caching schemes are adopted for nodes of different areas.
- We adopt a heuristic mechanism to implement on-path caching in the user-edge area and use a lightweight mechanism based on path competition to select one node in the non-edge area as the migration node to implement content migration of the user-edge area.
- We use real-world network topologies to evaluate the performance and scalability of the proposed caching scheme. The result shows that our scheme has better performance in different network topologies.

The rest of this paper is organized as follows. Section 2 reviews the related research of the ICN caching scheme. Section 3 introduces our proposed scheme PSBC, including the cache placement strategy and content migration mechanism. In Section 4, we carry out simulation experiments and analyze the results and performance of PSBC. Finally, we conclude and look into future research.

2. Related Work

Cache placement strategy is the subject of much research. Chai et al. [12] proposed a selection algorithm based on node centrality, in which the content is cached on the node with the highest betweenness centrality. In [13], the author proposed an auction-based caching, which allows the nodes on the data return path to coordinate caching decisions, with minimal overhead, through auctioning their “right to cache”. Ming et al. [14] proposed an age-based cooperative caching that aims at spreading popular content to the network edge based on lightweight collaboration mechanisms. In [15], the author proposed a caching strategy that considers nodes’ social attributes and selects nodes with high social attribute scores to cache contents.

The above approaches may lead to a situation where the content is too concentrated in a few nodes, which is not conducive to the full utilization of the network’s overall storage resources. One solution to this problem is to select content for migration. Luo et al. [16] proposed that the cache node can coordinate with neighbor nodes and migrate part of the content to the appropriate node in case of insufficient cache space. In [17], a cooperative caching policy is proposed, in which the on-path nodes optimally cache the requested content and are supported by a central off-path cache node for an additional level of caching. However, there is an additional communication overhead when choosing the migration node, and for Ref. [17], how to allocate the off-path cache node in a larger network is also a problem.

More ideally, Ref. [18] introduces a cache manager to get a global view of all network states for ideal cache management. However, in the case of a high dynamic ICN network, it is not feasible to use this approach for obtaining the status accurately and timely. Li et al. [19] proposed a cooperative caching strategy called cooperative in-network caching (CIC). In CIC, content is divided into different chunks and is cached at more than one node, and CIC adds a cooperative router table and a cooperative content store to keep track records of the location and identity of all cached chunks. However, because the ICN cache networks are highly dynamic and the cached content is updated, this type of collaboration

may incur additional communication overhead. A summary of the presented caching strategies in terms of their contributions is expressed in Table 1.

Table 1. Summary of contributions of the ICN caching strategies.

Caching Strategy	Contributions
CL4M [12]	Improving the caching gain by using the concept of betweenness centrality
BidCache [13]	Reducing caching redundancy by allowing nodes on the path to participate in an auction to win the right to cache the content
ABC [14]	Pushing popular content to the user edge adaptively by using “age” to control a content replica’s lifetime
SNCS [15]	Reducing caching redundancy by selecting influential nodes as cache nodes
NCBIC [16]	Improving the utilization of cache resources by content migration
Cooperative on-path and off-path caching [17]	Reducing user access latency by employing heuristic algorithms on the edge routers and the additional central router, respectively
Distribute cache management [18]	Achieving efficient caching by adopting a distributed autonomic management architecture
CIC [19]	Improving the utilization of cache resources by storing different kinds of chunks

3. Path Segmentation-Based Hybrid Caching

In this section, we first introduce the network architecture, then we describe path segmentation and the caching schemes in different areas.

3.1. IP-Compatible ICN Network Architecture

The existing Internet architecture is entrenched and challenged to change radically, and so the new architecture should be deployed as an overlay network to accommodate existing IP networks [20]. In this paper, we consider an IP-compatible ICN network architecture shown in Figure 1. IP network compatibility is achieved by extending the existing network layer and transport layer to support ICN functions. In this network architecture, the name resolution system (NRS) is an important network component aside from the routers that support caching. The NRS is often used for overlay network design, the name resolution process, and the decoupled content routing process [21]. In the network, content publishers and cache nodes register the mapping of IP addresses and content names with the NRS. ICN routers can process both IP streams (IP-based routing) and ICN streams (caching decisions).

Figure 1. IP-compatible information-centric network (ICN) network architecture.

3.2. Caching Idea Based on Path Segmentation

The user may be multiple hops away from the content source or cache node, and executing the same caching scheme for each node on the path does not obtain good caching

performance. We chose to divide each data return path into two segments—one segment is near the requesting user, called the user-edge area, and the other is relatively far from the user edge, called the non-edge area. Unlike node area division based on network topology information [8], this division of each return path is dynamic and temporary. We divide the nodes on the data return path according to the data transmission distance. The longer the data transmission distance is, the closer the node is to the user's edge.

Our research goal is to coordinate the two cache targets of user access latency and network resource utilization. User-edge area and non-edge area are biased towards different cache targets. To reduce user access latency, nodes in the user-edge area should cache the content with high popularity. We propose using the non-edge area as the secondary cache of the user-edge area. The content replaced in the user-edge area is migrated to one migration node chosen in the non-edge area. In order to avoid the communication overhead of selecting the migration node, when data are transmitted in the non-edge area, the nodes on the path collaborate in a competitive way to choose the most suitable node to be added to the data packet as the migration node of the user-edge area.

According to our mechanism, there are two types of network data packets: a normal data packet and a migrated data packet. We need to add several fields to carry some information in the header of the normal data packet and one field in the header of the migrated packet to implement this hybrid caching based on path segmentation. In order to reduce the transmitted data and network traffic overhead, related compression techniques can be adopted to compress the length of the packet header overhead, such as the small set of allowed coefficients (SSAC) algorithm [22].

The hop_count field: Since the network architecture we are considering is IP-compatible, the request and data transmission distance (hops) can be calculated by the Time To Live field (TTL). The hop_count is initialized to the distance between the hit node and the requesting user. The proportion of path segmentation is $\frac{\text{hop_count}}{k} : \left(\text{hop_count} - \frac{\text{hop_count}}{k}\right)$; we set $k = 2$ for the convenience of calculation in this paper, and the proportion can be optimized as a follow-up study. As shown in Figure 2, when the data transmission distance is less than $\frac{\text{hop_count}}{k}$, the node belongs to the non-edge area. Otherwise, the node belongs to the user-edge area.

Figure 2. Path segmentation.

The cache_pressure and migrate_node_address fields: These two fields are used to record the minimum cache pressure value and the corresponding node address in the non-edge area on each data return path, respectively. The calculation of cache pressure value and the migration node's selection in the non-edge area will be discussed later.

The popularity field: This field is used by both migrated and normal data packets. In this paper, we present the concept of caching value. Each cache node calculates the cache value for the incoming content and updates the cache value for the hit content; caching decisions on incoming content and the replacement and migration of cached content are determined by cache value. The popularity of the content is a major factor in determining the size of the cache value. We choose the number of times the content is requested in the cache node as the content's popularity. The *popularity* field is used to record the popularity of the content delivered by the packet at the source node of the packet.

3.3. Caching in User-Edge Area

Each cache node in the user-edge area of the data return path makes the caching decision individually. To achieve better caching, each node i in the user-edge area can consider the following optimization problem when deciding on caching:

$$\text{MAX } E[h_i] \quad \text{s.t. } E[d_i] < \delta \quad (1)$$

Or

$$\text{MIN } E[d_i] \quad \text{s.t. } E[h_i] > \delta \quad (2)$$

$E[h_i]$ is the average hit probability of the cache node i while serving the requests. $E[d_i]$ is the average end-to-end latency. To improve cache performance, we consider two common performance metrics: user access latency and cache hit ratio. For the optimal caching placement, each cache node i should ensure the caching of content, leading to low user access latency and high cache hit probability.

In our proposal, to achieve a more practical solution, we can focus on optimizing one goal and relaxing the optimization of another.

To implement the above optimization problem for each cache node i , we consider calculating the cache value $CV_i(k)$ for each incoming content k . The higher the popularity, the closer the content is to the user, and the more noteworthy it is to be cached. We maintain a cache value for each content cached by the node. The cache value of the hit content is updated when the user request hits on this node. In addition, cache replacement and migration are carried out according to the cache value of the cached content. To prevent cache space from being occupied by high cache value content that has not been accessed for a long time, we set a time period for each cached content. Expired content that is not accessed will be expelled. According to the mechanism, the following formula is used to calculate the cache value $CV_i(k)$ of each content k :

$$CV_i(k) = f1 * D_i(k) + f2 * P_i(k) \quad (3)$$

where $f1$ and $f2$ are weighting constants, $D_i(k)$ is the data transmission distance normalized by the distance between the requesting user and the hit node, and $P_i(k)$ is the content popularity normalized with the maximum popularity p_{max} of cached content. If the current node is a hit node, the cache value of the hit content is updated to $f2 * \left(P_i(k) + \frac{1}{p_{max}} \right)$.

The process of cache management in the user-edge area is described in Algorithm 1. The cache value is calculated for each incoming content (Step 2). We check the remaining cache space. When the cache space is insufficient, we select the content with the lowest cache value to replace, and we migrate the replaced content to the migration node carried by the packet (Steps 3–9).

Algorithm 1: Caching in User-Edge Area

```

current node:  $i$ 
remaining cache space:  $B_i$ 
1: for each incoming content  $k$ 
2:  $CV_i(k) = f1 * D_i(k) + f2 * P_i(k)$ 
3: if  $B_i == 0$  then
4:   if  $CV_i(k) < \min\{CV_i(n) \forall n\}$  then
5:     do not cache  $k$ 
6:   else
7:     find  $m$  where  $CS_i(m) == \min\{CS_i(n) \forall n\}$ 
8:     content  $m$  is replaced with  $k$ 
9:     forward  $m$  to  $k$ . migrate_node_address
10:  end if
11: else
12:  cache content  $k$ 
13: end if
14: if any content expires then
15:  expel the content
16: end if

```

3.4. Caching in Non-Edge Area

Nodes in the non-edge area receive two types of data packets: a normal data packet and a migrated packet. When a normal data packet is transmitted in a non-edge area, the nodes in the non-edge area collaboratively select a node as the user-edge area's migration node and add the node address to the packet. When the migration node receives the migrated packet, due to the node's limited cache space, the migration node should use an appropriate caching mechanism to decide whether to cache the migrated content.

3.4.1. Migration Node Selection

We select the migration node by calculating and comparing the cache pressure value CPV_i for each cache node i in the non-edge area. When calculating the cache pressure value for the cache node, we consider three parameters: cache space utilization, cache utilization, and cache replacement rate. Cache space utilization is the main indicator of the node cache status. The cache spaces of most cache nodes are close to saturation when the network is stable; because of this, we combine the cache space utilization with the cache replacement rate (the ratio of the content replaced to the total cache in a period) to measure the node cache status, and at the same time, we combine the cache utilization (the ratio of the hit content to the total cache in a period) to calculate cache pressure value CPV_i . The calculation formula of the cache pressure value CPV_i of node i is as follows:

$$CPV_i = \frac{UA(i)}{OP(i) + RP(i)} \quad (4)$$

where $UA(i) = \frac{\sum_j^{n_{hit}} c(j)}{CS(i)}$, $OP(i) = \frac{\sum_j^{n_{cache}} c(j)}{CS(i)}$, and $RP(i) = \frac{\sum_j^{n_{replace}} c(j)}{CS(i)}$ represent cache utilization, cache space utilization, and cache replacement rate, respectively. $CS(i)$ is the total cache size, and $c(j)$ is the size of each content.

When the data packet passes from the hit node to the first node in the non-edge area, its address and cache pressure value are added to the data packet. As described in Algorithm 2, the subsequent cache node in the non-edge area calculates its cache pressure value and compares it with the cache pressure value recorded in the data packet, which updates the corresponding information of the packet record if smaller. In this way, when the data packet is delivered to the user-edge area, the packet will record the node address that is most suitable for migration. This path competition-based migration node selection

avoids additional communication overhead relative to other migration node selection mechanisms [16–18].

Algorithm 2: Migration Node Selection

```

current node:  $i$ 
1: for any incoming content  $k$ 
2:  $CPV_i = \frac{UA(i)}{OP(i)+RP(i)}$ 
3: if  $CPV_i < k.cache\_pressure$  then
4:    $k.cache\_pressure = CPV_i$ 
5:    $k.migrate\_node\_address = i$ 
6: end if

```

3.4.2. Caching in the Migration Node

In Section 3.3, we mentioned that we maintain a cache value for each cached content, and the placement of incoming content, the replacement, and migration of cached content are all decided based on cache value. Therefore, the caching decision made by the migration node j on migrated content m should also be based on this mechanism. The calculation of cache value $CV_j(m)$ for the incoming migrated content should be different. We mainly consider the popularity of the content, and the calculation formula is as follows:

$$CV_j(m) = f2 * P_j(m) \quad (5)$$

where $f2$ is the weighting constant and $P_i(k)$ is the content popularity normalized with the maximum popularity p_{max} of cached content. As described in Algorithm 3, the migration node j calculates the cache value for each incoming migrated content (Step 3) and makes caching decisions according to the cache value. When the replacement occurs, the replaced content is directly expelled (Steps 4–9).

Algorithm 3: Caching in the Migration Node

```

current node:  $j$ 
remaining cache space:  $B_j$ 
1: for any incoming migrated content  $m$ 
2: if content  $m$  not in  $j$  then
3:    $CV_j(m) = f2 * P_j(m)$ 
4:   if  $B_j == 0$  then
5:     if  $CV_j(m) < \min\{CS_j(n) \ \forall n\}$  then
6:       do not cache  $m$ 
7:     else
8:       find  $c$  where  $CS_j(c) == \min\{CS_j(n) \ \forall n\}$ 
9:       content  $c$  is replaced with  $m$ 
10:    end if
11:  else
12:    cache content  $m$ 
13:  end if
14: end if
15: if any content expires then
16:   expel the content
17: end if

```

4. Performances Evaluations

We carried out a preliminary simulation of PSBC, and the results show the characteristics of our mechanism and its performance under different network topologies. We will introduce the simulation environment, experimental setup, and simulation results in detail in the following parts.

4.1. The Simulation Environment

We used the Icarus simulator [23] to run the simulation. Icarus is a Python-based ICN cache simulator used to simulate and evaluate the caching performance of various caching strategies in the ICN network environment. It could efficiently complete simulations consisting of millions of requests in a few minutes, and it supports users to implement their designed caching scheme and simulation.

4.2. The Experimental Setup

We chose some more typical cache placement strategies to compare with our scheme, such as LCE, LCD [24], ProbCache [25], CL4M [12]. For the caching strategies used for comparison, we chose LRU as the cache replacement algorithm. In our experiment scenario, to verify the scalability of PSBC in different network topologies, we chose the following real-world network topologies: GEANT (European academic network), WIDE (Japanese academic network), and GARR (Italian academic network). The topologies are shown in Figure 3. In the three topologies, nodes of degree 1 were set as receivers.

Figure 3. Simulation topologies.

According to our optimization objectives, we used two performance metrics (KPIs), i.e., cache hit ratio and average access latency, to measure the performance of the strategies. Cache hit ratio measures the portion of content requests served by a cache, and average access latency measures the delay taken to deliver content.

The selection of experimental parameters is shown in Table 2. In our simulation, we set the amount of content in the network to 3×10^5 , which is a reasonable amount of content in network topology and can be used to simulate real scenes. We set the number of warm-up requests to 3×10^5 and the number of requests tested to 6×10^5 . We assumed that the user's requests follow the Poisson distribution at a rate of 10 requests per second. The cache size for each node is the same. Cache size ratio reflects the total network cache size as a percentage of the total content, and we set its range to [0.001, 0.002, 0.004, 0.01] for simulation. We assume the content requests follow the Zipf popularity distribution, and the parameter skewness (α) of Zipf distribution is set to the range [0.2, 0.4, 0.6, 0.8, 1.0].

Table 2. Experiment parameters.

Parameter	Values
N_CONTENTS	3×10^5
N_WARMUP_REQUESTS	3×10^5
N_MEASURED_REQUESTS	6×10^5
REQ_RATE	10 req/s
CACHE_SIZE_RATIO	[0.001, 0.002, 0.004, 0.01]
SKEWNESS(α)	[0.2, 0.4, 0.6, 0.8, 1.0]

4.3. Simulation Result

In this part, we present the results of the simulation experiment. Each experiment was performed five times, and the result was an average of five experiments. Based on

the simulation results, a summary of the contrastive caching strategies in terms of their limitations is expressed in Table 3.

Table 3. Summary of limitations of the contrastive caching strategies.

Caching Strategy	Limitations
LCE	Causing cache redundancy and underutilization of cache space
LCD	Causing excessive eviction operations and maximum utilization of bandwidth
ProbCache	Differences in content popularity are not considered
CL4M	Causing load imbalance between nodes and underutilization of cache space

4.3.1. Impact of Skewness (α)

We change the size of skewness (α) to observe the impact on cache hit ratio and average access latency in three network topologies. In this part of the experiment, the cache size ratio is set to 0.04. The results are shown in Figure 4.

Figure 4. Cache performance with different skewness values (α).

A larger skewness (α) means more requests from users for popular content. As shown in Figure 4a–c, the cache hit ratio increases with skewness (α). The cache hit ratio is related to network resource utilization [8]. In the three network topologies, LCE causes cache redundancy, and the hit ratio of LCE is the lowest. CL4M caches content on nodes with the greatest betweenness centrality on the data return path, which results in an uneven distribution of content, so the cache hit ratio of CL4M is also relatively low. LCD caches content in the next-hop node of the hit node, and ProbCache caches content probabilistically according to data transmission distance and caching space available on the return path. The hit ratios of these two strategies are relatively close. PSBC has the highest cache hit ratio because popularity is the main factor for PSBC's caching decision, and the content migration mechanism improves the utilization of network resources. For example, when skewness (α) is 0.8, PSBC's cache hit ratio in GEANT topology is 48%, 40%, 18%, and 22%

higher than LCE, CL4M, PROB_CACHE, and LCD, respectively. As shown in Figure 4d–f, the average access latency decreases as skewness (α) increases. In PSBC, nodes close to users focus on caching popular content. We make a temporary area division for each data return path, which takes different users' access locations into account. The results show that the average access latency of PSBC is lower than the other four caching strategies under different skewness values (α).

4.3.2. Impact of Cache Size Ratio

We changed the cache size ratio to see the impact on cache hit ratio and average access latency in three network topologies. In this part of the experiment, skewness (α) is set to 0.6. The results are shown in Figure 5.

Figure 5. Cache performance with different cache size values.

The results show that PSBC performs better than the other four caching strategies in both cache hit ratio and average access latency. The increased cache size means that more contents can be cached. As shown in Figure 5a–c, the cache hit ratio increases as the cache size ratio increases in the three network topologies. LCE has the lowest cache hit ratio due to cache redundancy. LCD reduces cache redundancy relative to LCE and potentially considers the content's popularity, with the cache hit ratio being second only to PSBC in three topologies. ProbCache makes caching decisions based on data transmission distance and the cache space available on the return path but ignores differences in content popularity. CL4M cannot effectively utilize network cache resources as the cache size is increased. The cache hit ratio of these two caching strategies is lower than PSBC. PSBC takes both network resource utilization and content popularity into account. With the increase of cache size ratio, PSBC has a more obvious advantage in cache hit ratio. In Figure 5d–f, we observe the cache size ratio's impact on the average access latency. To reduce user access latency, nodes near users should cache as much popular content as possible. In PSBC, nodes in the user-edge area mainly consider the popularity of content when making caching decisions. The non-edge area is used as a secondary cache of the user-edge area to improve cache resource utilization. This mechanism gives PSBC an advantage. Nodes near the user's edge can cache more popular content as the cache size increases, which greatly reduces the average access latency.

5. Conclusions

In this paper, we proposed PSBC, a hybrid caching scheme based on path segmentation. The proposed scheme divides each data return path into the user-edge area and non-edge area according to the data transmission distance. The two areas adopt different caching schemes to coordinate user access latency and network resource utilization. The user-edge area implements optimal caching decisions to provide faster responses for users. The non-edge area is used for content migration from the user-edge area to improve network resource utilization. The simulation results show that the proposed scheme is better than other popular caching strategies regarding the cache hit ratio and access latency. In future work, we plan to optimize the path segmentation method further, propose a more flexible path segmentation algorithm, and apply the caching scheme to other application scenarios for testing.

Author Contributions: Conceptualization, W.L., P.S., and R.H.; methodology, W.L., P.S., and R.H.; software, W.L.; writing—original draft preparation, W.L.; writing—review and editing, W.L., P.S., and R.H.; supervision, P.S.; project administration, R.H.; funding acquisition, P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by Strategic Leadership Project of Chinese Academy of Sciences: SEANET Technology Standardization Research System Development (Project No. XDC02070100).

Acknowledgments: We would like to express our gratitude to Peng Sun, Rui Han, Li Zeng, and Yuanhang Li for their meaningful support for this work.

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

References

1. Pan, J.; Paul, S.; Jain, R. A survey of the research on future internet architectures. *IEEE Commun. Mag.* **2011**, *49*, 26–36. [\[CrossRef\]](#)
2. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 26–36. [\[CrossRef\]](#)
3. Koponen, T.; Chawla, M.; Chun, B.-G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A Data-Oriented (and beyond) Network Architecture. In Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007; Association for Computing Machinery: New York, NY, USA, 2007; pp. 181–192.
4. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.; Briggs, N.; Braynard, R. Networking Named Content. *Commun. ACM* **2012**, *55*, 117–124. [\[CrossRef\]](#)
5. Trossen, D.; Parisi, G. Designing and Realizing an Information-Centric Internet. *IEEE Commun. Mag.* **2012**, *50*, 60–67. [\[CrossRef\]](#)
6. Dannewitz, C.; Kutscher, D.; Ohlman, B.; Farrell, S.; Ahlgren, B.; Karl, H. Network of Information (NetInf)—An information-centric networking architecture. *Comput. Commun.* **2013**, *36*, 721–735. [\[CrossRef\]](#)
7. Zhang, G.; Li, Y.; Lin, T. Caching in information centric networking: A survey. *Comput. Netw.* **2013**, *57*, 3128–3141. [\[CrossRef\]](#)
8. Zhang, G.; Wang, X.; Gao, Q.; Liu, Z. A Hybrid ICN Cache Coordination Scheme Based on Role Division between Cache Nodes. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015; pp. 1–6.
9. Din, I.U.; Hassan, S.; Khan, M.K.; Guizani, M.; Ghazali, O.; Habbal, A. Caching in Information-Centric Networking: Strategies, Challenges, and Future Research Directions. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1443–1474. [\[CrossRef\]](#)
10. Ioannou, A.; Weber, S. Towards on-path caching alternatives in Information-Centric Networks. In Proceedings of the 39th Annual IEEE Conference on Local Computer Networks, Edmonton, AB, Canada, 8–11 September 2014; pp. 362–365.
11. Nguyen, X.N.; Saucez, D.; Turlatti, T. Efficient caching in Content-Centric Networks using OpenFlow. In Proceedings of the 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Turin, Italy, 14–19 April 2013; pp. 67–68.
12. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache “less for more” in information-centric networks (extended version). *Comput. Commun.* **2013**, *36*, 758–770. [\[CrossRef\]](#)
13. Gill, A.S.; D’Acunto, L.; Trichias, K.; Brandenburg, R.V. BidCache: Auction-Based In-Network Caching in ICN. In Proceedings of the 2016 IEEE Globecom Workshops (GC WKSHPS), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
14. Ming, Z.; Xu, M.; Wang, D. Age-based cooperative caching in information-centric networking. In Proceedings of the 2014 23rd International Conference on Computer Communication and Networks (ICCCN), Shanghai, China, 4–7 August 2014; pp. 1–8.
15. Yanqing, C.; Muqing, W.; Min, Z.; Kaili, W. Socially-Aware NodeRank-Based Caching Strategy for Content-Centric Networking. In Proceedings of the 2016 International Symposium on Wireless Communication Systems (ISWCS), Poznan, Poland, 20–23 September 2016; pp. 297–303.

16. Luo, X.; An, Y. Neighbor Cooperation Based In-Network Caching for Content-Centric Networking. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 2398–2415.
17. Rath, H.K.; Panigrahi, B.; Simha, A. On Cooperative On-Path and Off-Path Caching Policy for Information Centric Networks (ICN). In Proceedings of the 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), Crans-Montana, Switzerland, 23–25 March 2016; pp. 842–849.
18. Sourlas, V.; Gkatzikis, L.; Flegkas, P.; Tassiulas, L. Distributed Cache Management in Information-Centric Networks. *IEEE Trans. Netw. Serv. Manag.* **2013**, *10*, 286–299. [[CrossRef](#)]
19. Li, Z.; Simon, G. Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching. In Proceedings of the 2011 IEEE International Conference on Communications (ICC), Kyoto, Japan, 5–9 June 2011; pp. 1–6.
20. Shailendra, S.; Panigrahi, B.; Rath, H.K.; Simha, A. A novel overlay architecture for Information Centric Networking. In Proceedings of the 2015 21st National Conference on Communications (NCC 2015), Bombay, India, 27 February–1 March 2015.
21. Liao, Y.; Sheng, Y.; Wang, J. A Survey on the Name Resolution Technologies in Information Centric Networking. *J. Netw. New Media* **2020**, *9*, 1–9.
22. Gligoroski, D.; Kravetska, K.; Overby, H. Minimal Header Overhead for Random Linear Network Coding. In Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW), London, UK, 8–12 June 2015; pp. 680–685.
23. Saino, L.; Psaras, I.; Pavlou, G. Icarus: A Caching Simulator for Information Centric Networking (ICN). In Proceedings of the SIMUTools '14: 7th International ICST Conference on Simulation Tools and Techniques, Lisbon, Portugal, 17–19 March 2014.
24. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634. [[CrossRef](#)]
25. Psaras, I.; Chai, W.K.; Pavlou, G. Probabilistic In-Network Caching for Information-Centric Networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking, Helsinki, Finland, 17 August 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 55–60.