applied sciences

Article

An NDN Cache Management for MEC

DaeYoub Kim [1] and Jihoon Lee [2,*]

[1] Department of Information Security, Suwon University, Hwaseong-si, Gyeonggi-do 18323, Korea; daeyoub69@gmail.com
[2] Department of Electronic and Information System Engineering, Sangmyung University, Cheonan-si, Chungcheongnam-do 31066, Korea
* Correspondence: vincent@smu.ac.kr; Tel.: +82-41-550-5411

Received: 25 October 2019; Accepted: 20 January 2020; Published: 29 January 2020

Featured Application: Future Internet Architecture, Multi-Access Edge Computing.

Abstract: To enhance network performance, the named data networking architecture (NDN) caches data-packets in the network nodes on a downstream network path. Then it uses such cached requested data-packets to respond to new request-packets. Hence, a cache management scheme (CMS) is the essential point of NDN. CMS generally considers two main factors. One is a short response time and the other is storage efficiency. To rapidly respond to requests, CMS generally tries to cache data-packets near users as much as possible. To efficiently manage storage, it uses the popularity of the data. That is, proportionally to the popularity of the data, it increases the number of nodes caching data-packets and manages the lifetime of caches. However, few data objects are as popular as many users globally enjoy in the real world. Hence, if the assumptions about content- usage are practically changed, CMS can waste cache storage and not significantly improve network efficiency. We show that many caches have expired and are not used at all. To improve such inefficiency of CMS, this paper propose to simultaneously apply two cache decision factors, the expected frequency of a cache hit and the popularity of data. That is, it proposes to gradually cache transmitted data in nodes in which their expected cache-usage frequency is relatively high. To show the effectiveness of our proposal, we implement LdC (a limited domain cache policy) and evaluate the performance of LdC. The evaluation result shows that it can enhance the cache-storage efficiency by up to 65% compared with existing CMS without degrading the network efficiency.

Keywords: P2P; MEC; ICN; NDN; cache management scheme

1. Introduction

Since P2P networking technology was first introduced, it has become the general approach of advanced network technologies to use caches (i.e., replica of data), which saves multiple nodes. One of the main characteristics of such technologies is to shift the networking paradigm from 'host-centric networking' to 'content-centric networking' [1–3]. When using in-network caches, data-providers can expect that other nodes can handle most request-packets and users can seamlessly retrieve requested data regardless of the network and system state of the data-providers. Hence, it appears very attractive to utilize caches from the viewpoints of users and data-providers.

The named data networking architecture (NDN), which is a one information centric networking architecture (ICN), caches a data-packet (*Data*) in the network-nodes on a downstream network path of the *Data*. When a node receives a request-packet (*Interest*) for the *Data*, if the node has cached the *Data*, it directly transmits the cached *Data* to requesters as a response. Since one of the intermediate nodes on an upstream network path of *Interest* can directly respond to the *Interest*, a user can receive the

Data more rapidly. Hence, NDN is considered as one of alternative solutions to efficiently implement mobile edge computing [4].

Theoretically, NDN is designed to cache *Data* on all nodes of a downstream network path of the *Data*. However, considering the amount of data that is transmitted through networks today, it must be assumed that each network-node requires huge storage space. Additionally, if each network-node on the upstream network path of *Interest* tries to search the relevant *Data* in its huge storage, it can cause serious service-delays. Therefore, various cache management schemes have been presented to improve this inefficiency of NDN [5–10].

The cache management scheme (CMS) generally tries to reduce unnecessary cache redundancy and to improve transmission overheads for transmitting *Data*. To do this, many CMSs utilize common approaches as follows:

- To gradually increase the number of nodes caching *Data* proportionally to the request frequency of the *Data*.
- To cache *Data* as near a requester-side edge network (RsEN) as possible.

In particular, the latter approach expects that *Interest* generated in RsEN can be responded by nodes in the RsEN without transmitting the *Interest* to the core network. To do this, it should be assumed that 'many' nodes of 'many' RsENs have cached relevant *Data*. However, it would not be a practical assumption because most content objects are not as popular as expected. For example, the average usage ratio of YouTube content does not exceed 0.01%. Such an analysis leads to two facts:

- Many *Interests* that are generated in RsENs can be transmitted to the original content providers.
- Even if *Data* is cached in several RsENs, except for a few famous content objects, the caches of most *Data* cannot be used at all. Therefore, such caches can just waste the storage of nodes.

In this paper, to improve the storage overheads of existing CMS, we propose an improved scheme considering two factors at the same time as follows:

- The popularity of the *Data*. As existing CMSs, we also gradually increase the number of nodes caching *Data* proportionally to the generation frequency of the *Interests* requesting the *Data*.
- The cache usage rate of each node. We first analyze the cache hit rate of nodes. Then, we select nodes such that the cache hit frequency of these nodes is relatively high.

When applying these factors, this proposed scheme can enhance the storage efficiency of nodes up to 65% compared with existing CMSs without degrading the network efficiency.

2. Named-Data Networking Architecture

2.1. NDN Overview

NDN uses a hierarchical data name that uniquely identifies *Data*. Based on this unique data name of *Data*, NDN selects an upstream network path through which the *Interest* will be forwarded toward the publisher of the *Data*. Additionally, NDN can efficiently search a matched cache for the *Interest* in the network using a data name. NDN divides the content into several segments and then generates *Interests* for each segment. Figure 1 shows the NDN *Interest*/*Data* forwarding procedure.

① Requestor-A transmits *Interest* for a segment of data.

② When a node (Node-b) receives the *Interest* through the incoming network interface (Face-b1), Node-b reads a lookup table (PIT, Pending Interest Table). An entry of the PIT consists of both the data name of the *Interest* and the incoming network interface (Face) of the *Interest*. Node-b conducts the PIT lookup to find a PIT entry matching the *Interest*. If there is a match, Node-b adds Face-b1 to the found PIT entry and then finishes handling the *Interest*. Otherwise, Node-b records a new entry on the PIT for the *Interest*.

③ Node-b conducts the CS (Content Store) lookup using the data name of the *Interest* to find a matched cache to the *Interest*. If there is a match, Node-b forwards the found cache to Requestor-A through Face-b1 and then deletes the relative entry from the PIT. It then finishes dealing with the *Interest*.

④ Otherwise, Node-b forwards the *Interest* using the information from the FIB (Forwarding Information based) table.

⑤ If a node having the relevant *Data* receives the *Interest*, the node forwards the *Data* along the reverse of the path through which the *Interest* passed.

⑥ When receiving the *Data*, Node-b conducts the PIT lookup to find a PIT entry matching the *Data*. If there is no matching entry, Node-b discards the *Data*.

⑦ Otherwise, Node-b caches the *Data* in its CS, and then forwards the *Data* through the network interfaces that are recorded on the found PIT entry.

⑧ Finally, Requestor-A receives the *Data*.

⑨ Thereafter, Requestor-B generates *Interest* for the *Data* that has been previously transmitted to Requestor-A.

⑩ If Node-b receives the *Interest* from Requestor-B, Node-b forwards the cached *Data* to Requestor-B and then accomplishes the operation of the *Interest*.

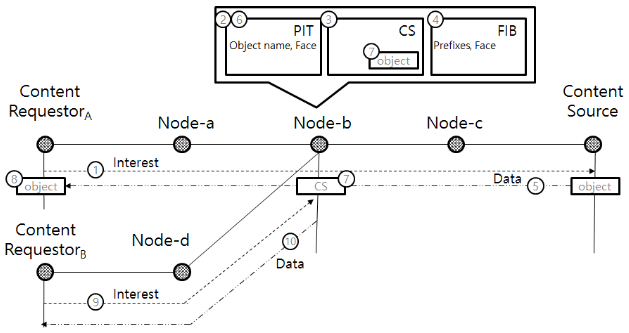


Figure 1. Data networking architecture (NDN) *Interest/Data* Procedure.

2.2. NDN Cache Management Overview

Considering the transmission overheads for forwarding *Interest/Data*, the storage overheads for caching *Data*, and the computation overheads for searching the matched *Data* in-network caches, many CMSs proposed to selectively and gradually cache transmitted *Data* propositionally to the request frequency of the *Data* [5–11]. Some CMSs probabilistically/randomly select one or several nodes on a downstream path of the *Data* and then cache the *Data* in the selected nodes [5–7]. Such schemes can reduce the unnecessary redundant caches. However, they cannot correctly predict/evaluate the transmission overheads because they can neither predict nor designate which nodes on a downstream path will cache transmitted *Data*.

Other CMSs gradually increase the number of nodes caching *Data* propositionally to the request frequency of the *Data* [8–11]. They can enhance the transmission overheads and reduce the unnecessary cache redundancy [12–14]. Both LCD (Leave Copy Down) and WAVE take this approach. Figure 2 shows WAVE [10]. If the *k*th node on an upstream path of *Interest* has cached the matched *Data* and it forwards the *Data*, LCD/WAVE proposes that only the first node receiving the *Data* caches the transmitted *Data*. Practically, the node is the *k-1*th node on the upstream path of the *Interest*. While LCD increases the number of newly cached segments of content by one for each time the content is required, WAVE exponentially increases the number of segments that are newly cached, as shown in Figure 2.

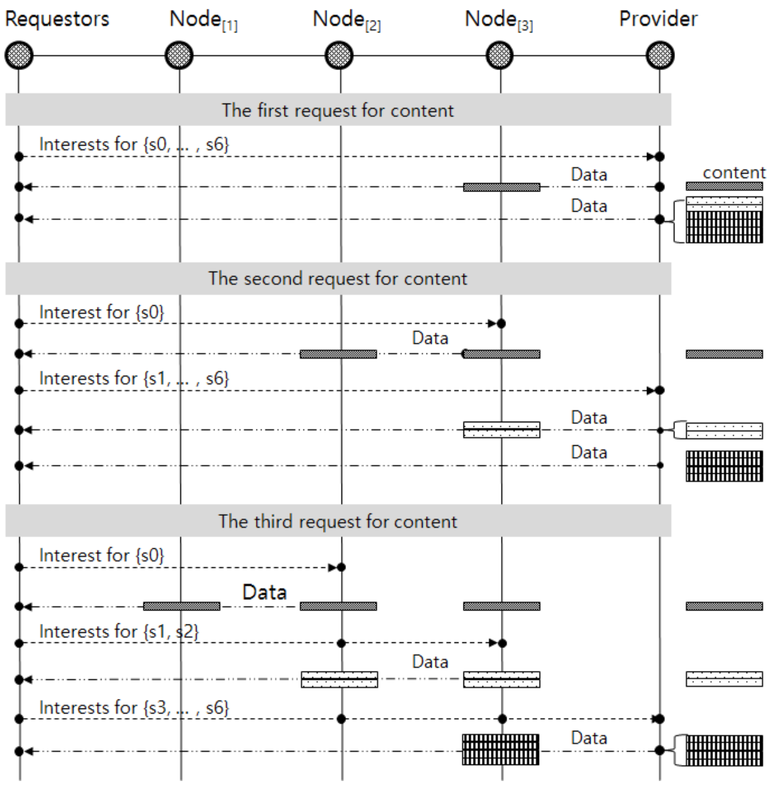

Requestors
Node[1]
Node[2]
Node[3]
Provider
The first request for content
Interests for {s0, ... , s6}
Data
content
Data
The second request for content
Interest for {s0}
Data
Interests for {s1, ... , s6}
Data
Data
The third request for content
Interest for {s0}
Data
Interests for {s1, s2}
Data
Interests for {s3, ... , s6}
Data

Figure 2. WAVE Procedure.

(1) Let the content consist of seven segments $\{s_0, \ldots, s_6\}$. When receiving the first *Interests* of the content, only the first segment s_0 of the content will be cached in a node (Node$_{[3]}$) on a downstream path. Other segments are transmitted to a requestor without being cached.

(2) If the second *Interests* for the content are forwarded, Node$_{[3]}$ can respond to the *Interest* for requiring the cached s_0. Then, while the s_0 is transmitted, only Node$_{[2]}$ caches it. The other *Interests* for $\{s_1, \ldots, s_6\}$ are responded to by the provider of the content. Then, only two segments, s_1 and s_2, are newly cached in Node$_{[3]}$.

(3) Finally, if the provider receives the third *Interests* for the content, four segments $\{s_3, \ldots, s_6\}$ are newly cached in Node$_{[3]}$.

2.3. NDN Cache Management Limitation

Most CMSs commonly propose to cache popular content on RsEN. Such schemes expect that many *Interests* generated in RsEN can be responded to within the RsEN. So, they also expect that the number of *Interest/Data* transmitted to the core network can be reduced. However, such an expectation has some unrealistic issues.

First, such schemes generally assume that the requested content is popular such that many users who have geographically dispersed locations in RsENs enjoy them. However, that is a very unusual case. For example, YouTube statistics published in 2019 show that more than 1.9 billion users watch YouTube content a month, 250 million hours of YouTube viewed per day on TV screens, and the most popular videos generate over 3 billion views. However, considering the average cumulative views of each video, most videos generate less than 10,000 views [14–18]. This means that most YouTube videos are viewed only by 0.0005% of users. Hence, the number of RsENs that have cached the requested content may be smaller than expected. In this case, most *Interests* will finally be transmitted to content providers. Additionally, although some contents have been cached in RsENs, most caches in RsENs may not be used if the cache hit rate of the nodes in RsENs is relatively low. Moreover, if a cache hit rate of caches is low, the caches can be deleted from the RsEN over time due to the cache management policy of the RsEN provider.

Second, such schemes overlook a practical fact that the providers of RsENs do not have any responsibility to cache the transmitted content. Instead, as shown in the case of the CDN (Content Delivery Network) service, it is generally accepted that content providers pay a cost for caching content. Hence, it cannot be guaranteed that RSEN providers cache the received content generated by publishers who are not members of their RsENs.

3. Cache Decision Factors

To solve the problems mentioned in the previous section, this paper proposes two decision factors as follows:

- The first factor is the popularity of the *Data*. As existing CMSs, it proposes to gradually increase the number of nodes caching Data proportionally to the generation frequency of the *Interests* requesting the *Data*.
- The second factor is the cache usage rate of each node. For that, we first analyze the cache hit rate of the nodes. Figure 3 shows the evaluation result of the cache usages of NDN. To evaluate the cache usages of NDN, we utilize a network topology and configuration as described in Section 5. This assumes that the average popularity of the content is 0.001. It means that a new *Interest* for the content is published per each 1000 s. Then it measures the cache hit rate of each node according to the different cache lifetimes (1 h, 1 day, and 1 week). Specifically, if a custodian based-NDN is applied [19], the cache hit rate of the custodian nodes of a publisher-side domain is relatively higher than those of RsENs. Figure 3 shows that the cache usages rate of the nodes of the domain of a publisher is relatively high.

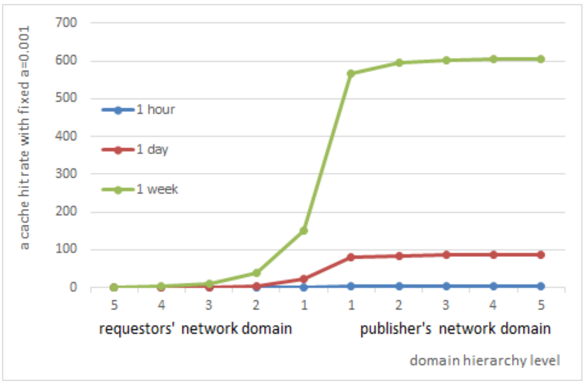


Figure 3. The cache hit rate of NDN node comparison.

Figure 3 could cause misunderstanding that caching *Data* in RsEN is useless. However, the result of Figure 3 has some limitations because the average popularity of the content is fixed. Nevertheless, it is sufficiently meaningful since it shows the necessity of consideration of the cache usage rate of a node.

4. A Limited Domain Cache Management Policy

In this section, we propose a simple cache policy (a limited domain cache policy, LdC) using the previously described two factors. It is the main purpose of this proposal to prove the usefulness of applying the previous two caching decision factors to CMS simultaneously.

For that, we select nodes such that the cache hit frequency of these nodes is relatively high. LdC aims at 'gradually' caching *Data* in nodes that many *Interests* are 'frequently' passed through. Also, like LCD/WAVE, it gradually increases the number of nodes caching *Data*. As shown in Figure 3, since nodes in a publisher-side network domain have a higher hit rate than nodes in a requestor-side network domain, we simply suggest improving LCD such that it caches the transmitted *Data* only in the nodes of the publisher-side network domain. Then we compare the performances between existing CMSs and LdC to show the effect of the two factors.

4.1. Hierarchical Named Node

NDN originally used a hierarchical data name. Recently, although NDN additionally utilizes a flat data name like other ICNs, NDN does not utilize a name resolution server that resolves the flat name of the content to a host identity caching the content. Therefore, to forward *Interest* for *Data* to a content provider which generated the *Data*, NDN still needs a hierarchical data name for the *Interest*.

Taking advantage of the characteristics of a layered data name, is an efficient approach to transmit the *Interest* by referring to the prefixes of the data name of the Interest sequentially from a higher layer prefix. Therefore, the LdC rationally assumes that there are hierarchical representative nodes to which *Interests* having the same name prefixes are commonly forwarded. That is, these nodes are the domain gateway nodes of a publisher-side network domain. The nodes have a hierarchical node identity which publishers use as the prefix of the data name. So, every *Interest* having a data name in which the prefix of the data name is the same as the node identity should be transmitted to one of these nodes. For example, *Interest* having a data name "/n1/n2/n3/text.txt/s1/239" is first forwarded to a node having the identity "/n1". Then the node forwards the *Interest* to an internal node having the identity "/n1/n2". The LdC calls such nodes the representative nodes of the name prefix "/n1". A node can have several different identities. LdC assumes that the data name hierarchy of NDN consists of two parts, as shown in Figure 4.

- A named domain hierarchy (NDH): NDH is used to forward *Interest* toward a data publisher. In Figure 4, "/suv/research/security/david" is the NDH. A name prefix with 1-layer is "/suv". Then, one of the representative nodes having identity "/suv" handles the *Interest* with the data name prefix "/suv". In addition, a name prefix with a 2-layer is "/suv/research". One of the representative nodes having identity "/suv/research" handles the *Interest* with the data name prefix "/suv/research".
- Object File Information (OFI): This is the information of a segment of data. It is formatted as "/doc/report.txt/s2/239". The "/doc" denotes the file path of the segment. The "/report.txt" is the file name of the segment. The "s2" is the segment number and the "239" is the time stamp.

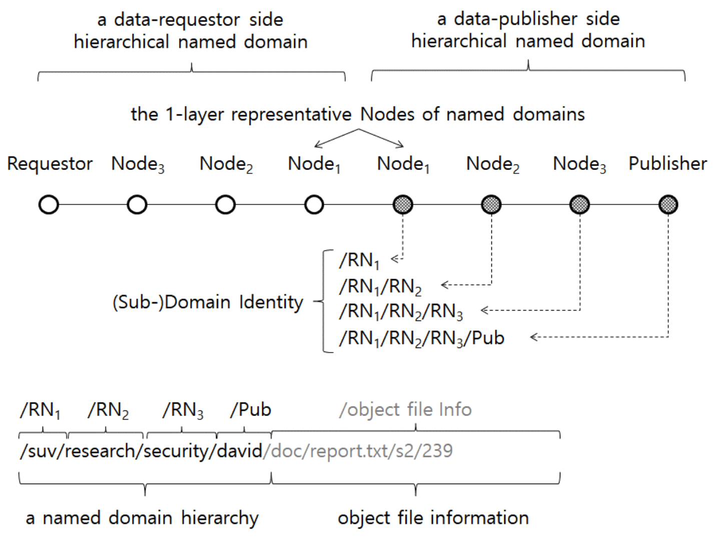

a data-requestor side hierarchical named domain
a data-publisher side hierarchical named domain
the 1-layer representative Nodes of named domains
Requestor
Node$_3$
Node$_2$
Node$_1$
Node$_1$
Node$_2$
Node$_3$
Publisher
(Sub-)Domain Identity
/RN$_1$
/RN$_1$/RN$_2$
/RN$_1$/RN$_2$/RN$_3$
/RN$_1$/RN$_2$/RN$_3$/Pub
/RN$_1$
/RN$_2$
/RN$_3$
/Pub
/object file Info
/suv/research/security/david/doc/report.txt/s2/239
a named domain hierarchy
object file information

Figure 4. Data name hierarchy.

4.2. Limited Domain Caching Policy (LdC)

For simple implementation, LdC utilizes a custodian-based NDN [19]. Each network domain has a custodian node that every *Interest*/*Data* entering or leaving this domain must pass through. LdC proposes to utilize the custodian node as a representative node. By this assumption, when data is published in a named domain, all *Interests* for the data are commonly forwarded via the custodian node of the domain. Hence, as shown in Figure 4, if a node having the identity "/suv/research/security/david" receives the *Interest* for *Data* having the data name "/suv/research/security/david/doc/report.txt/s2/239", the *Data* are forwarded to the custodian nodes (i.e., representative nodes) of the domains named "/suv/research/security", "/suv/research", and "/suv" in order. These representative nodes gradually cache the *Data* like LCD/WAVE.

To limit a cache area to a publisher-side name domain, if the representative node of a named domain receives *Data* with a name prefix that does not match its node identity, the node forwards the Data without caching the Data. To implement this functionality, LdC adds two fields, a cache flag (CF), and a cache remove flag (RF), to the *Data* structure.

- If the CF of the *Data* has '1' and the domain name of a node receiving the *Data* matches the name prefix of the *Data*, the node caches the *Data*. Otherwise, the node does not cache it.
- The RF is an optional field. It is used to delete the content that is cached in a node after the node forwards the *Data*. If the RF has '1', except for the custodian node with the layer 1 name domain of a data publisher, each node deletes the cached *Data* after forwarding the *Data*.

Figure 5 describes in detail the LdC procedure. Assume that the CF of the *Data* has '1'.

① If a node receives the *Interest* for *Data* that it has cached, the node forwards the *Data* back to the requestors. Then, if the node is neither a custodian node of a name domain nor the publisher of the *Data*, it checks whether the RF of the *Data* has '1'. If it does, the node deletes the cached *Data* from its CS.

② If a node receives *Data* from its neighboring node, the node confirms the CF of the *Data*. If CF = 0, it forwards the *Data* without caching the *Data*. Otherwise, the node compares its node name with the name domain hierarchy of the *Data*. If these two names match, it caches the *Data* with CF = 1. Otherwise, it does not cache the *Data*.

③ The node sets the CF of the Data to '0'. Then, it sends the *Data* to the requestors. However, it still has a cache of the *Data* that has a CF = 1.

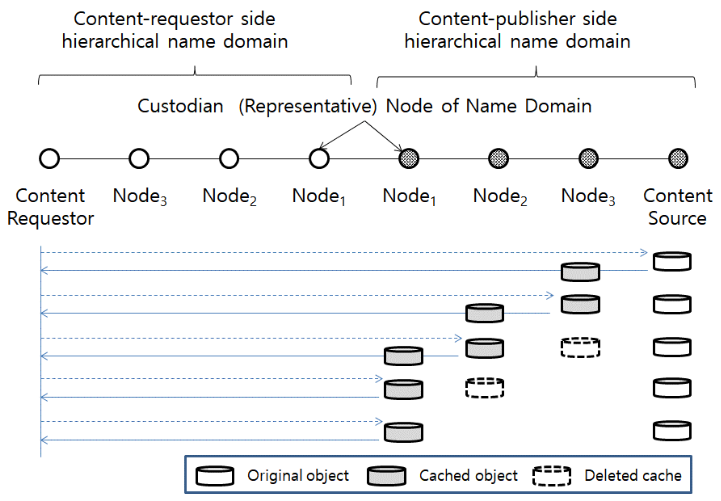

Content-requestor side hierarchical name domain
Content-publisher side hierarchical name domain
Custodian (Representative) Node of Name Domain
Content Requestor
Node3
Node2
Node1
Node1
Node2
Node3
Content Source
Original object
Cached object
Deleted cache

Figure 5. LdC (a limited domain cache policy) procedure using custodian-based NDN.

Using LdC, the cache of *Data* gradually moves toward a node matching the highest layer name prefix of the *Data*. Additionally, to enhance the performance of LdC, it is possible to increase the amount of cached *Data* per each content transmission time like in WAVE.

5. Performance Evaluation

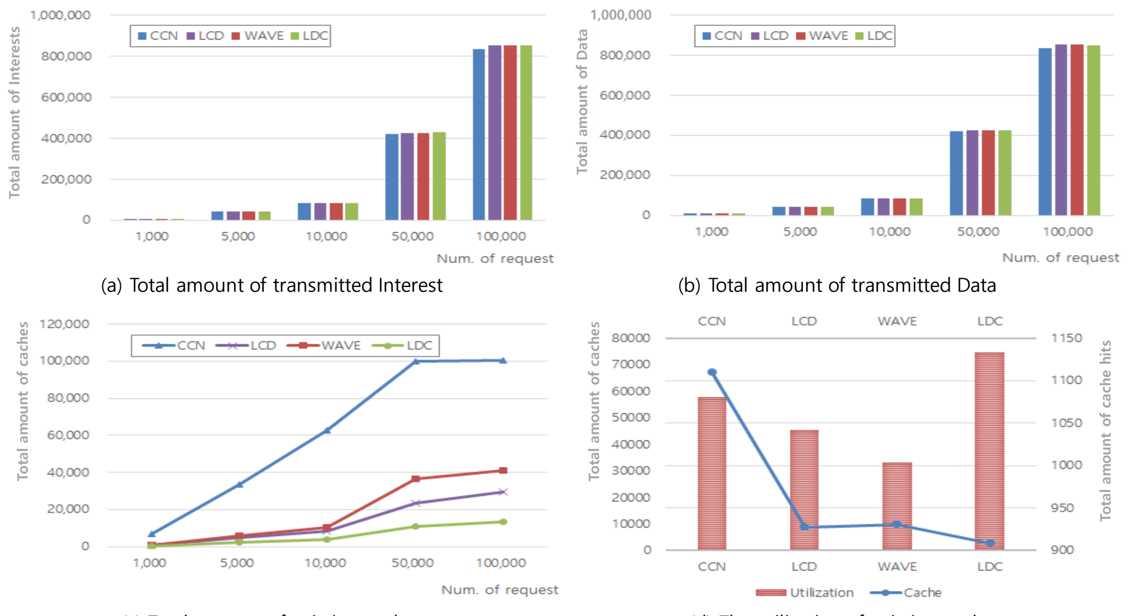
To evaluate the performance using a simulation (an ns-3 based ndnSIM), we configure a discrete event-driven simulation and utilize a name domain topology with five domains as follows:

- A domain consists of a hierarchical name domain topology with a depth of 5.
- A root node has 5 child nodes. Except for leaf nodes, each internal node has 4 child nodes, respectively.
- A root node is a custodian node of the domain.
- Except for a root node and leaf nodes, each node is the custodian node of a hierarchical subdomain.
- A leaf node is a user's device.

Additionally, the configuration for this simulation is as follows.

- 16,000 objects are available, and these objects have the same popularity.
- Each node can cache up to 1000 objects.
- A node has a cache life of 50 s. Therefore, if no cache hit occurs for 50 s, the cache is deleted from the node.
- We randomly selected users' devices to generate the *Interests* for objects that are also randomly selected.

Figure 6 shows the results of the performance evaluation of the four schemes: [NDN], [LCD], [WAVE], and [LdC]. "The number of requests" means the number of *Interests* generated by users. The average response rate is 96.7%. Figure 6a,b shows the total number amount of *Interests* and Data respectively which are transmitted between nodes. It shows that, in the three cases of the [LCD], [WAVE] and [LdC], the number of transmitted *Interests* increases a small amount. However, this difference is very slight. It means that the hop counts of the transmitted Interests/Data are similar when applying these four schemes. Additionally, it means that almost the same nodes respond to the *Interests* regardless of the applied scheme.



(a) Total amount of transmitted Interest (b) Total amount of transmitted Data

(c) Total amount of existing caches (d) The utilization of existing caches

Figure 6. Performance simulation result.

Nevertheless, as shown in Figure 6c, there are huge differences among the storage overheads of these four schemes. It is expected that [LdC] improves the storage overheads for caching Data by approximately 65% compared with [LCD]/[WAVE] and by approximately 95% compared with [NDN]. Hence, Figure 6a–c shows that LdC can improve the storage overheads of nodes without the degradation of transmission performance.

Furthermore, Figure 6d shows the comparison result of the cache utilization of these four schemes. When randomly generating 9996 *Interests*, we analyze the total amount of cached Data in the network (represented by lines) and count the number of cache hits (represented by bars). Although [LdC] caches much less Data than other schemes, [LdC] cache utilization is at least 10% to at most 50% higher.

6. Conclusions

Since NDN utilizes the in-network caches of content, a cache management scheme is an essential technology of NDN. So far, most cache management schemes cache transmitted content in RsENs. Because of this, such an approach would make it possible to reduce the amount of network traffic that is exchanged between core networks. Such an approach assumes that the content's popularity is related to how frequently it is requested. However, many reports have shown that the usage rate of most content shared through the Internet is low. Hence, although content has been cached on nodes, it can happen that the caches of the content are not used at all. Therefore, existing CMSs for NDN can waste the cache storage of a node but do not significantly improve network efficiency. Practically, when we analyze both the *Interest*/*Data* traffic flow of NDN and the cache usage rate of each node, we find that some caches are not utilized at all. Also, we find that *Data* that is cached in the requestor- side networks has a lower hit rate than *Data* that is cached in the provider-side network.

Hence, we propose to consider simultaneously two decision factors, both a content popularity and the cache hit rate of each node, when NDN selects nodes to cache transmitted *Data*. Also, to prove the effectiveness of our proposal, we suggest a simple cache model using our proposal and evaluate the performance of the model. Since this proposed scheme caches *Data* on nodes with high cache hit rates, it enhances the cache-storage efficiency by up to 65% compared with existing cache management schemes in which the content is cached both in the requestor-side network and in the provider-side network without degrading the network efficiency. That is, LdC improves the storage overheads of nodes without degradation of transmission performance.

The main contribution of this work consists of three points: First, we showed that the cache hit rate of a node is an important factor as much as the popularity of content. When NDN considers these two factors simultaneously, it can improve the storage overheads of nodes.

Second, when implementing MEC for caching content in edge networks, it is necessary to consider the case where caches in edge networks are not used at all. Hence, it should be needed to design a special cache management scheme for requestor-side network domain.

Finally, we suggest a future work which sets the cache lifetime of each node considering these two factors. CMS generally considers the lifetime of caches. However, most CMSs take a formalized rule and use a fixed value. Using these two factors, it may be possible to generate a flexible lifetime of caches.

Author Contributions: Conceptualization, D.K. and J.L.; methodology, D.K.; software, D.K.; validation, D.K. and J.L.; formal analysis, D.K.; investigation, D.K.; resources, D.K.; data curation, D.K.; writing—original draft preparation, D.K.; writing—review and editing, J.L.; visualization, J.L.; supervision, D.K.; project administration, D.K.; and funding acquisition, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1B03034215 and 2016R1D1A1B03931628).

Conflicts of Interest: The authors declare no conflict 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. Jacobson, V.; Smetters, D.; Thornton, J.; Plass, M.; Briggs, N.; Braynard, R. Networking Named Content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome, Italy, 1–4 December 2009; pp. 1–12.
3. The NDN Project Team. Named Data Networking (NDN) Project. In *NDN Technical Report NDO-0001*; The NDN Project Team: Wood, CA, USA, 2010.
4. Amadeo, M.; Campolo, C.; Molinaro, A.; Rottondi, C.; Verticale, G. Securing the Mobile Edge through Named Data Networking. In Proceedings of the IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 5–8 February 2018.
5. Psaras, I.; Chai, W.K.; Pavlou, G. In-network cache management and resource allocation for information-centric networks. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *25*, 2920–2931. [CrossRef]
6. Laoutaris, N.; Che, H.; Stavrakakis, I. The lcd interconnection of lru caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634. [CrossRef]
7. Zhang, G.; Li, Y.; Lin, T. Caching in information centric networking: A survey. *Comput. Netw.* **2013**, *57*, 3128–3141. [CrossRef]
8. Zhang, M.; Luo, H.; Zhang, H. A Survey of Caching Mechanisms in Information-Centric Networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1473–1499. [CrossRef]
9. Luo, Z.; LiWang, M.; Lin, Z.; Huang, L.; Du, X.; Guizani, M. Energy-Efficient Caching for Mobile Edge Computing in 5G Networks. *Appl. Sci.* **2017**, *7*, 557. [CrossRef]
10. Cho, K.; Lee, M.; Park, K.; Kwon, T.; Choi, Y.; Pack, S. WAVE: Popularity-based and Collaborative In-network Caching for Content-Oriented Networks. In Proceedings of the IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN), Orlando, FL, USA, 25–30 March 2012.
11. Aloulou, N.; Ayari, M.; Zhani, M.F.; Saïdane, L. A popularity-driven controller-based routing and cooperative caching for named data networks. In Proceedings of the 6th International Conference on the Network of the Future (NOF), Montreal, QC, Canada, 30 September–2 October 2015.
12. Shailendra, S.; Sengottuvelan, S.; Rath, H.K.; Panigrahi, B.; Simha, A. Performance evaluation of caching policies in NDN—An ICN architecture. In Proceedings of the 2016 IEEE Region 10 Conference (TENCON), Singapore, 22–25 November 2016.
13. Yanuar, M.R.; Manaf, A. Performance evaluation of progressive caching policy on NDN. In Proceedings of the International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA), Denpasar, Indonesia, 16–18 August 2017.
14. Newberry, E.; Zhang, B. On the Power of In-Network Caching in the Hadoop Distributed File System. In Proceedings of the ACM Conference on Information Centric Networking, Macau, China, 24–26 September 2019.
15. Clement, J. YouTube—Statistics & Fact. 2019. Available online: https://www.statista.com/topics/2019/youtube/ (accessed on 29 December 2019).
16. How Many Views Does A Youtube Video Get? Average Views by Category. Available online: https://tubularinsights.com/average-youtube-views/ (accessed on 29 December 2019).
17. Mansoor Iqbal. YouTube Revenue and Usage Statistics 2019. Available online: https://www.businessofapps.com/data/youtube-statistics/#4 (accessed on 29 December 2019).
18. Zhi, W.; Wenwu, Z.; Shiqiang, Y. *Online Social Media Content Delivery: A Data-Driven Approach*; Springer-Verlag New York Inc.: New York, NY, USA, 2018.
19. Jacobson, V.; Braynard, R.L.; Diebert, T.; Mahadevan, P.; Mosko, M.; Briggs, N.H.; Barber, S.; Plass, M.F.; Solis, I.; Uzun, E.; et al. Custodian-based Information sharing. *IEEE Commun. Mag.* **2012**, *50*, 38–43. [CrossRef]