2021 the 7th International Conference on Computer and Communications

Limit-Caching: A Caching Scheme Based on Limited Content Popularity in ICN

2021 7th International Conference on Computer and Communications (ICCC) | 978-1-6654-0950-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCC54389.2021.9674587

Hongyu Guo
Shenzhen Graduate School
Peking University
Shenzhen, China
guo_hongyu@pku.edu.cn

Hui Li*
Shenzhen Graduate School
Peking University
Shenzhen, China
lih64@pkusz.edu.cn

He Bai
Shenzhen Graduate School
Peking University
Shenzhen, China
baihe@pku.edu.cn

Tao Sun
Network Information Center
University Town of Shenzhen
Shenzhen, China
suntao@utsz.edu.cn

Zhe Huang
University Town Library of Shenzhen
University Town of Shenzhen
Shenzhen, China
huangzhe@utsz.edu.cn

***Abstract*—The explosive growth of network devices and content poses huge challenges to the current host-centric Internet architecture. Information-centric network (ICN) proposed a promising content-oriented network paradigm. The in-network caching in ICN reduces the content retrieval latency and network traffic as well as enhances the overall network performance. Therefore, cache management in ICN has attracted wide attention of researchers recently. But a caching scheme with lower network traffic and better adaptability remains to be studied. In this paper, we propose Limit-Caching, a caching scheme based on limited content popularity in ICN. In this scheme, we use tokens to control the insertion rate of content and encourage content providers to set high popularity levels for popular content, and routers probabilistically insert caches based on popularity levels. At the same time, the content that is repeatedly requested will be gradually pushed to the edge of the network. We also design Least Recently Used with Time (LRU-T) based on LRU replacement policy to adjust the caching probability. We conduct extensive experiments with a tree topology to evaluate our scheme. Simulation results show that Limit-Caching has better performance than several classic ICN caching schemes under multiple metrics.**

Keywords—Information-centric network, caching strategy, content caching, popularity level

I. Introduction

Today's Internet is experiencing explosive growth in the number of connected devices and generated content. Users no longer care about whether the content provider has established a connection but rather the content itself. At the same time, users can both consume content and generate content, which makes the Internet more complicated. Recently, the Information-centric network (ICN) has emerged as a novel network paradigm designed to provide a receiver-driven many-to-one content distribution, which is different from the current host-centric IP network. There are a variety of typical ICN instances, including Multi-Identifier network (MIN) [1], named data network (NDN) [2], and NetInf [3], etc. The design details of these architectures are different, but they all address content by content name. ICN includes cache management modules in all routers. Each intermediate node in the network can selectively store the passing content replicas. When the same content is requested again, it can be retrieved directly from the cache of the router rather than the original content provider.

The emergence of ICN decouples the content provider from the content, putting the content and the content provider in separate locations [4]. The in-network caching in ICN reduces the server load of content providers and response time of user requests, which improves the overall network throughput. However, the size of the cache in the router is much smaller than the storage capacity of the host, so how to cache data has become a topic worthy of research in ICN. If all content arriving at the router is cached, it will cause unimaginable storage redundancy. In addition, cache invalidation caused by a large number of cache insertions will also dramatically reduce the cache hit ratio. To make the cache more efficient, the cache space must be used in a planned way.

To solve the above problems, we propose Limit-Caching: a caching scheme based on limited content popularity in ICN. The main modules of our scheme include 1) *Token mechanism*: The router calculates the token generation and consumption of each content provider based on the content popularity level. 2) *Push-down mechanism*: The router records the number of times the content is requested until the content is replaced. When the number of requests exceeds the threshold, the content will be pushed to the next router closer to the user. 3) LRU-T: LRU-T, an improved LRU strategy, will record the time that each content stays in the cache space before being replaced out of the cache space then counts the average time that each content provider's content stays in the cache and dynamically adjusts the content caching probability. To evaluate our cache structure, we perform a lot of simulation experiments on ndnSIM [5]. Experiments show that Limit-Caching can perceive the changes in content popularity compared to other solutions. It effectively improves the cache hit ratio and reduces the average number of hops and server load.

The rest of this paper is organized as follows. Section II reviews related works, Section III describes the design and implementation details of Limit-Caching, and Section IV shows the performance evaluation of simulation experiments. Section V summarizes the paper.

II. Related Works

In the research of in-network caching in ICN, researchers have made great efforts to improve cache efficiency.

The Leave Copy Everywhere (LCE) caching strategy proposed by V. Jacobson *et al.* [6] is a prevalent caching strategy applied in the ICN caching scheme. LCE will cache every packet that passes through the router. This scheme is straightforward, but LCE does not consider the impact of content popularity and network topology. its shortcoming lies in frequent cache replacement operations. The Leave Copy Down (LCD) method proposed by Laoutaris *et al.* [7] will move the data to the next node closer to the user when the cache hits, which helps to gradually push the popular content to the edge. LCD takes into account the influence of network topology on the cache in the network, and this method implicitly considers the popularity of the content. However, the convergence speed of LCD is slow, and the popular content will not reach the edge node until after many moves. In [8], W.K. Chai *et al.* adopted a scheme of caching content only at nodes of maximum betweenness centrality, and proposed Betweenness method, which effectively improved the caching capacity of the network, but this scheme did not consider the popularity of the content, and the nodes of maximum betweenness centrality will become the bottleneck of network. The cache replacement strategy proposed by Zheng *et al.* [9], betweenness and edge popularity (BEP), is an improvement on the betweenness method.

III. System Model

A. Design Overview

TABLE I. List Of Symbols Used In The Paper

Symbol	Description
$T_{r,cp}$	Number of tokens of content provider cp on router r
H_r	Cache hit rate of router r
K_r	Content movement threshold on router r
Δt	Time interval of scheduled tasks
θ_r	average time for each content to be replaced on router r
$\theta_{r,cp}$	average time for content provider cp 's content to be replaced on router r
$\varphi_{r,cp}$	The offset of popularity level of content provider cp on router r
N_{op}	Number of insert and lookup operations on the cache
$Interest(c)$	Interest packet for content c
$Data(c)$	Data packet for content c

There will always be unpopular content cached in routers in ICN networks, which may not be requested by users until they are replaced after being cached. If the router caches such content in large quantities, it will cause frequent cache failures and a decrease in cache hit ratio. In order to alleviate the above two problems, we proposed Limit-Caching. In the Limit-Caching architecture, each content is assigned a popularity level by the content provider, which is represented by $Level_c$. When the content passes through the router, the router caches the content and records that the content is requested before being replaced. We have designed a *token mechanism*, a *push-down mechanism*, and an LRU-T cache replacement strategy, which allows more popular content to be cached on routers closer to users, and reduces the copy of the content on the network. TABLE I. summarizes the most used symbol in the paper.

Since Limit-Caching needs to carry the content popularity level and support *push-down mechanism*, we partially modified the data packet in the classic ICN. In the data packet, we added two fields $Move$ and $PopularityLevel$. The $Move$ field is a *boolean* variable. If the content will be pushed to the edge because the number of requests exceeds the threshold, it is set to $True$; otherwise, it is set to $False$. $PopularityLevel$ is an *integer* variable used to store the popularity level of the content.

In our model, we assume that all routers in the network have the same computing power as well as the same cache size. Also, to simplify the model, we believe that all content in the network has the same size. Taking into account the content service model of the Internet, we assume that the requests sent by users satisfy the Poisson distribution. The popularity of content satisfies the Zipf-like distribution [11].

B. Limit-Caching Mechanism

T-Caching[10] is a scheme that uses tokens to control the cache insertion rate, which significantly reduces the amount of content inserted in the cache and increases the cache hit rate. The content in T-Caching is divided into two categories: cacheable and non-cacheable. Content providers can get tokens when they decide not to cache content, and they need to consume tokens when they decide to cache content. This approach encourages content providers to actively cache highly popular content, which helps to improve cache hit rates. However, this caching method does not significantly improve in terms of reducing the average number of hops, and it is too rough to classify content as only cached and not cached. In order to address these issues, we have made some improvements on the basis of T-Caching and proposed the Limit-Caching algorithm.

Algorithm 1 Algorithm for adjusting β

```
Variable: $\varepsilon \leftarrow$ variation of $\beta$
1:  $cnt \leftarrow 0$
2:  for every $\Delta t$ do
3:      $h \leftarrow cache\ hit\ times$
4:      $m \leftarrow cache\ miss\ times$
5:      if $h/(m+h) > H_r$ then
6:          $cnt \leftarrow cnt + 1$
7:      else
8:          $cnt \leftarrow 0$
9:          $\varepsilon = -1 \cdot \varepsilon$
10:     end if
11:     $\beta = \beta + \varepsilon \cdot 1.1^{cnt}$
12:     $H_r = h/(m+h)$
13: end for
```

The content provider will assign a popularity level from 0 to 100 to each content, and the router will cache the content probabilistically based on the popularity level. In order to prevent too much low popularity content from being cached, we use a *token mechanism* to control the rate of cache insertion. Before the cache is inserted, it is necessary to ensure that the content provider has generated sufficient tokens on this router for the consumption of this insert operation, or the cache will not be allowed. The Limit-Caching scheme implies that content providers set low popularity content to a lower popularity level to acquire tokens so that high popularity content will be cached in the router with a higher probability. *Token mechanism* guarantees that popular content will be cached in the router and significantly reduces the number of content insertions. $T_{r,cp}$ is defined as (1)

$$T_{r,cp} = \sum_{c} f(Level_c), \quad (1)$$

where $f(Level_c)$ represents the function of token generation/consumption with respect to content popularity level, we let

$$f(Level_c) = \begin{cases} \beta \cdot \cos\left(\frac{Level_c}{100} \cdot \pi\right), 0 < Level_c \leq 50, \\ \cos\left(\frac{Level_c}{100} \cdot \pi\right), 50 < Level_c \leq 100 \end{cases}. \quad (2)$$

Algorithm 2 Algorithm for data packet processing

```
Input: $Data(c)$ is data packet of content $c$
 1:  $cp \leftarrow Data(c).topLevelName$
 2:  if $Data(c).MoveTag$ equals to $TRUE$ then
 3:      cache $Data(c)$      // push-down mechanism
 4:      $Data(c).SetMoveTag(FALSE)$
 5:      return
 6:  end if
 7:  $level \leftarrow Data(c).PopularLevel$
 8:  $t \leftarrow$ calculate token generation/consumption by (2)
 9:  if $T_{r,cp} + t \leq 0$ then
10:      return                  // Caching is not allowed
11:  end if
12:  $T_{r,cp} \leftarrow T_{r,cp} + t$        // calculate token
13:  $p \leftarrow level + \varphi_{r,cp}$       // plus probability offset
14:  cache $Data(c)$ with probability $p$
```

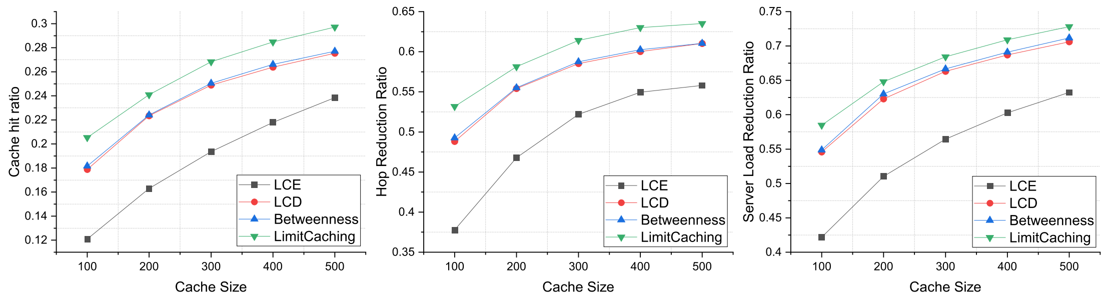
In (2), the β controls the rate of generation of $T_{r,cp}$. If the value of β is too large, unpopular content will be cached, causing more content to be replaced out of the cache. Similarly, if a too small value of β is used, only a small amount of content will be added to the cache, and it is not sensitive to changes in cache popularity, which will reduce the cache hit rate. The β sets too large or too small, and the cache hit rate will be reduced. Therefore, we dynamically adjust the β by monitoring the changing trend of the cache hit rate over a period of time. In order to speed up the convergence, we have optimized the changes in the same direction of β. The details are given in Algorithm 1.

When the router receives a data packet, the router first checks the $MoveTag$ of the data packet. If it is $TRUE$, it means that the packet is obtained from the cache of the previous router through the *push-down mechanism*, and the data packet must be cached. Otherwise, data packets will be cached probabilistically based on the popularity level only when the tokens are sufficient. Probabilistic caching effectively reduces the number of copies of content on the path.

In addition, we also designed an improved cache replacement strategy: LRU-T (LRU with time), which is on the basis of LRU. We will record the time that each data packet stays in the cache space before being replaced out of the cache space (to simplify the calculation, we use the number of operations instead of the timestamp), calculate the average time for each content to be replaced, and the average time for each content provider's content to be replaced. Every Δt, we compare the magnitude of θ_t and $\theta_{r,cp}$ to determine whether the popularity level set by the content provider cp is over or under, and thus adjust the popularity level offset. LRU-T penalizes content providers who give the fake popularity rating to the cache probability and encourages content providers to cache on demand. Detailed procession of data packet demonstrated in Algorithm 2, and the details of the caching process are shown in Algorithm 3.

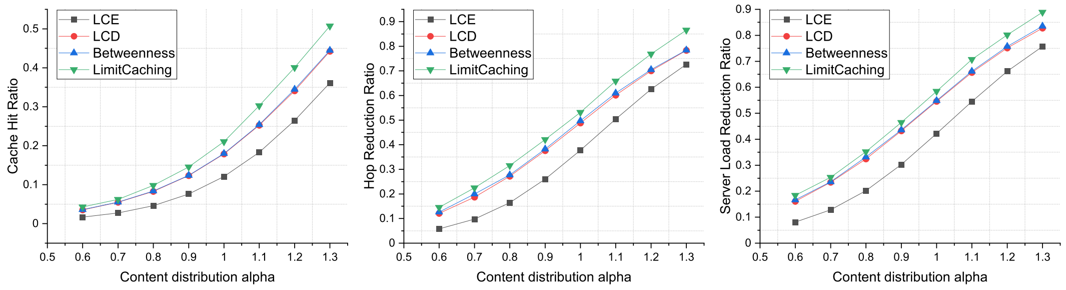
Algorithm 3 Algorithm for caching processing

```
Input: $Data(c)$ is the content $c$ to be cached
       Cache probability $p$
Variable: $time[c]$ records the insertion time of content
       $\epsilon \leftarrow$ variation of $\varphi_{r,cp}$
Phase 1: cache $Data(c)$
 1:  $time[Data(c)] \leftarrow N_{op}$
 2:  insert $Data(c)$ into cache space
 3:  $Data(r) \leftarrow replaced\ data\ from$ cache space
 4:  $rcp \leftarrow Data(r).topLevelName$
 5:  $stayTime \leftarrow N_{op} - time[Data(r)]$
 6:  $time[Data(r)] \leftarrow 0$
 7:  $\theta_r \leftarrow a \cdot \theta_r + b \cdot stayTime$
 8:  $\theta_{r,rcp} \leftarrow a \cdot \theta_{r,rcp} + b \cdot stayTime$
 9:  $N_{op} \leftarrow N_{op} + 1$
Phase 2: adjust offset
10:  for every $n$ operation do
11:      for every content provider $cp$ do
12:          if $\theta_r < \theta_{r,cp}$ then
13:              $\varphi_{r,cp} \leftarrow \varphi_{r,cp} + \epsilon$
14:          else
15:              $\varphi_{r,cp} \leftarrow \varphi_{r,cp} - \epsilon$
16:          end if
17:      end for
18:  end for
```



(a) α = 1.0 with different cache size (b) α = 1.0 with different cache size (c) α = 1.0 with different cache size

Fig. 1. Performance with different cache size



(a) cache size =100 with different α (b) cache size =100 with different α (c) cache size =100 with different α

Fig. 2. Performance with different α

Since the popularity of content will change over time, it is possible that it will become popular or unpopular. It is not enough to just use the initial popularity level set by the content provider. For this reason, we designed a *push-down mechanism* based on the LCD. While caching the content, it also records the number of times $Cnt_{r,c}$, the content was requested by the user before being replaced out of the cache space. If $Cnt_{r,c}$ exceeds the threshold K_r, $Cnt_{r,c}$ will be reset to $K_r/2$. This is done considering that there may be other forwarding paths that will also request this content. And before this content is forwarded to the next-hop router, its $Move$ field will be set to $TRUE$. The *push-down mechanism* ensures that popular content is cached in routers closer to the user.

IV. Performance Evaluation

We did a detailed experiment on ndnSIM [5]. The network topology is a k-ary tree with n layers. The arrival of requests from each user is subject to Poisson distribution. Each user generates 20 requests per second, and the cache size ranging from 100 to 500, and the popularity of the content is subject to the Zipf distribution with the parameter α, and the parameter α range 0.6 from 1.3. The parameters in the experiment are listed in TABLE II.

To evaluate the performance of Limit-Caching, we conduct experiments in the k-ary tree topology, in which we set the $level$ to 5, k to 2. All leaf nodes are a set of content requesters, the root node is the multiple content providers, and the remaining nodes are routers. The network topology is shown in Fig. 3.

We compared the following caching schemes: LCE, LCD, and Betweenness. We use three indicators to evaluate performance: cache hit ratio, hop count reduction, and server load reduction ratio.

- Cache hit ratio: The cache hit rate of the overall network topology. In the simulation, we calculated the overall hit rate of all routers.
- Hop count reduction: It is the number of routers that are reduced by the cache when users request content.
- Server load reduction ratio: The server load reduction ratio of the content provider. In the experiment, we calculated the percentage reduction in the frequency of user requests arriving at the server.

TABLE II. Parameters Settings

Parameter	Value
α range	0.7~1.3
Cache size range	100~500
Content number	10000
Content Size	1500 bytes
Warm-up request	100000
Height of tree	5
k	2
Request rate	20/s

The test first uses 100,000 requests for a warm-up and then starts collecting data after ensuring that the network has stabilized.

A. Cache Hit Ratio

From Fig. 1 (a), we can know that the cache hit rate, with each cache structure, will increase with the increase of cache size. As the cache size increases, more content can be cached in the network, which increases the cache hit rate. Our scheme has better performance for different cache size. Fig. 2 (a) shows the effect of α changes on the cache hit rate. Limit-Caching has better performance for different α. This is because other programs do not take good advantage of the popularity of the content.

B. Hop Count Reduction

It can be seen from Fig. 1 (b) that as the cache size increases, the average number of hops decreases because more content is stored on the router. Limit-Caching has better performance. This is because probability storage guarantees that popular content is stored in the router closer to the user, while the *push-down mechanism* shortens the distance between popular content and the user. Fig. 2 (b) shows that as the α increases, the hop reduction ratio also increases. As the α increases, the content distribution becomes more concentrated, more popular content is cached in the router, and more requests are satisfied at the intermediate router, thus reducing the average number of hops. With different cache sizes and different content distribution parameters α, Limit-Caching outperforms other schemes and greatly reduces the average number of hops for interest.

C. Server Load Reduction Ratio

From Fig. 1 (c), we can know that as the cache size increases, the load on the server gradually decreases. With different cache sizes, our scheme outperforms other schemes. This is because in our architecture, the high popularity content will be pushed to the router closer to the user, and the low popularity content will be cached on the router closer to the content provider. And probabilistic caching also reduces the number of times the same content is cached in the network. This scheme increases the diversity of content. At the same time, we can also see from Fig. 2 (c) that our scheme is better than other schemes with different α. This is because our scheme takes good advantage of the content's popularity.

V. Conclusion

In this paper, we propose a new type of ICN caching scheme, Limit-Caching, where content will be restrictedly cached in the router based on popularity, which effectively improves the efficiency of caching. In Limit-Caching, the *token mechanism* controls the rate of cache insertion. Each data packet will be preset by the content provider with a popularity level. If the popularity level is high, tokens will be consumed, and tokens will be generated if the level is low. Only when the content provider has sufficient tokens on the router, the data packet will be cached. At the same time, this scheme implements a *push-down mechanism* based on LCD, which pushes the content that has been requested multiple times to the router closer to the user. Moreover, this scheme also adopts the LRU-T caching strategy, which dynamically adjusts the caching probability of the content provider according to the time that the cache stays in the cache space when the cache is replaced. The scheme encourages content providers to assign popularity levels based on the actual distribution of content popularity. Experiments show that Limit-Caching outperforms several representative methods, get better performance in terms of cache hit ratio, hop reduction ratio and server load reduction ratio on in different scenarios. In the future, we plan to comprehensively evaluate the performance and scalability of Limit-Caching in different network topologies.

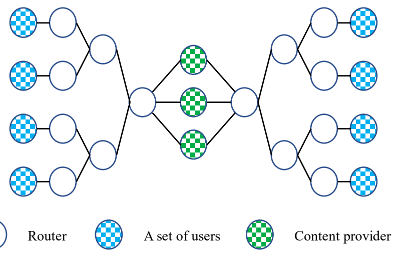


Fig. 3 Topology of the evaluated network

Acknowledgment

This work is supported by the Key-Area Research and Development Program of Guangdong Province under Grant 2019B010137001, Shenzhen Research Programs under Grant GXWD20201231165807007-20200807164903001, JCYJ202-10324122013036, JCYJ20190808155607340, the National Keystone Research and Development Program of China under Grant 2017YFB0803204, China Environment for Network Innovation under Grant GJFGW No. [2020]386, SZFGW [2019] 261.

References

[1] H. Li et al., "MIN: Co-Governing Multi-Identifier Network Architecture and Its Prototype on Operator's Network," in IEEE Access, vol. 8, pp. 36569-36581, 2020, doi: 10.1109/ACCESS.2020.2974327.

[2] L. Zhang, et al. "Named data networking (ndn) project." Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC 157 (2010): 158.

[3] Dannewitz, Christian, et al. "Network of information (netinf)–an information-centric networking architecture." Computer Communications 36.7 (2013): 721-735.

[4] B. Nour et al., "A Unified Hybrid Information-Centric Naming Scheme for IoT Applications," Computer Communications, vol. 150, pp. 103–114, 2019.

[5] A. Afanasyev, M. Ilya, and L. Zhang, "ndnSIM: NDN simulator for NS-3," Univ. California, Los Angeles, Tech. Rep. 4, 2012.

[6] V. Jacobson et al., "Networking named content," in Proc. CoNEXT Conf., 2009, pp. 1–12.

[7] N. Laoutaris et al., "The LCD interconnection of LRU caches and its analysis," Perform. Eval., vol. 63, no. 7, pp. 609–634, 2006.

[8] W. K. Chai et al., "Cache 'less for more' in information-centric networks (extended version)," Comput. Commun., vol. 36, no. 7, pp. 758–770, 2013.

[9] Q. Zheng et al., "A cache replication strategy based on betweenness and edge popularity in named data networking," in Proc. IEEE ICC Conf., 2019, pp. 1–7.

[10] S. Lee, I. Yeom and D. Kim, "T-Caching: Enhancing Feasibility of In-Network Caching in ICN," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 7, pp. 1486-1498, 1 July 2020, doi: 10.1109/TPDS.2020.297070

[11] Adamic L A, Huberman B A. Zipf's law and the Internet[J]. Glottometrics, 2002, 3(1): 143-150.