LeafPopDown: Leaf Popular Down Caching Strategy for Information-Centric Networking

*[1]Hizbullah Khattak, [2]Noor Ul Amin, [3]Ikram ud Din, [4]Insafullah, [5]Jawaid Iqbal
Department of Information Technology
[1,2,5]Hazara University Mansehra,
[3]University of Haripur,
[4]Abbotabad University of Sciences & Technology
K-P, Pakistan

***Abstract*—Information-Centric Networking is a name based internet architecture and is considered as an alternate of IP base internet architecture. The in-network caching feature used in ICN has attracted research interests as it reduces network traffic, server overload and minimizes latency experienced by end users. Researchers have proposed different caching policies for ICN aiming to optimize performance metrics, such as cache hits, diversity and eviction operations. In this paper, we propose a novel caching strategy of LeafPopDown for ICN that significantly reduces eviction operation and enhances cache hits and diversity in ICN.**

Keywords—Component; information-centric networking; caching; popularity; least recently used

I. Introduction

The data traffic and number of users of internet are growing rapidly during the last few years. Global IP video traffic will reach to 82% of all internet users' traffic by 2021 [1].

The ICN is an alternate network paradigm of traditional host based network communication model of internet [2]. The information in ICN is retrieved by name instead of its host locality identifier to provide data to the users with minimum delay. In order to achieve this aim, in-network caching is used in ICN to store the contents for easily access by the end users. The other features of multicast, routing by name and encryption support are included in ICN. The researchers have presented some ICN architectures such as Content Centric Networks [3], NetInf [4] and PURSUIT [5]. However, CCN have received more attention in the research community. Most of the research community focuses on designing efficient caching strategy for ICN as caching is the main characteristic of ICN.

The in-network caches are used in CCN storing several replicas of information in the network. The requests made in the future for these data can be served from these storages reducing access delay to the users and load on server and helps in minimizing the congestion in network. Researchers have proposed novel caching strategies for improving performance of CCN. Recent research work in ICN have identified that content popularity is an important factor in improving performance of ICN [6]. However, the existing policies fail to cache the content effectively so that these caches could be efficiently utilized in order to increase cache hit, diversity and reduce cache eviction operations. It is therefore very important to design caching policy that can enhance the cache utilization and avoid the content redundancy.

We proposed a LeafPopDown caching policy for ICN that cache the unpopular content near the end users and popular content on downward node and on leaf node. LeafPopDown calculate the popularity of content at each node and when content popularity increases from a specific threshold, it then caches the content on downward node and leaf node otherwise, only on leaf node. In this way, load on server or any specific node does not increase from a threshold level and redundancy in the networks is avoided.

This paper is organized as such: Section II discuss related work to our proposed caching policy. Section III discusses our proposed caching policy of LeafPopDown and its algorithm. In Section IV, we discuss the analysis and evaluation of LeafPopDown caching policy and Section V, we conclude our paper.

II. Related Work

Here, we discuss caching strategies related to our proposed caching strategy.

The simplest and default caching strategy for all the architectures of ICN are Leave Copy Everywhere [7], which cache object on each node of a data delivery path. Though this caching policy has an advantage of faster data dissemination however, it causes a huge redundancy and resource consumption to its alternatives.

In MPC [8], the authors calculate popularity of content in content popularity table locally at each node. When content popularity increases from specific threshold; content is cached on the neighbors' node. This caching policy has a drawback of storing content on the neighbor nodes of a serving node away from the nodes near the users.

The authors in [9] proposed a progressive caching policy, in which object is stored on one downstream node of hit node and on intermediate node of incoming links greater than threshold. This caching policy avoids storing the unpopular content. However, it shares the shortcoming of Leave Copy Down and fixed popularity caching because of its reliance on their functionality.

The caching policy of Breadcrumbs [10] is proposed to efficiently utilize off-path caches. After the arrival of content requests to the server, each router stores a pointer called breadcrumbs along the downloading path. This pointer shows the direction of the sent contents. When requests arrive for content, it encounters a breadcrumb and that breadcrumb redirects the request in that direction.

Cho et al. [11] proposed to segment the content and caching the chunks exponentially based on popularity of the content. The idea is to store the content progressively near users with increasing requests. In WAVE, the upstream node recommends its downward node to store the number of chunks by using caching suggestion flag bit in content reply's packet. If the flag bit is 1, chunk is cached otherwise not. WAVE has some limitation. It focuses on accessing the object request and hence it does not enhance the performances of network if the users are requesting a part of an object rather than full objects.

Badov et al. [12] proposed the caching-awareness to in-network caching. The aim of this caching policy is to reduce the download times to the users. CAC avoids using the congested links and storing the object on downstream end of congested link. This caching strategy is based on two factors, i.e., download time to the users and the content popularity and it is performed on every node of a delivery path. The caching capacity of network is considered 5% of the total content population and the Zipf popularity distribution (α = 0.8) is used. This caching policy outperforms in terms of average retrieval delay as compared to other caching policies. However, in case of average hit rate metric; it does not.

III. Proposed Caching Strategy

We assume ICN is a graph of $G = (V, E)$. In this graph, $V = (v_1 \dots \dots . v_n)$ is a group of nodes where each node is having limited storing capability and $E = (e_1 \dots \dots . e_m)$ represent links between these nodes.

We further assume that request for data follows design of Name Data Networking [2]. An INTEREST packet is forwarded for the desired content and that request is forwarded towards server till it finds the copy of the required content. The routing table is created in Forwarding Information Base (FIB) by OSPFN protocol. We further consider that routing nodes advertise fair information. In response of an INTEREST packet, data packet is delivered on the request traversing path by using the Pending Interest Table (PIT). For simplicity we assume here that the node have same cache size. For calculation of content popularity, we consider that each router count the number of request for content in particular time T.

We illustrate and compare workflow of our proposed caching strategy of LeafPopDown and LCE through an example.

The example is explained in Fig. 1.

Fig. 1(a) represents the general scenario of networks. The content in the network is stored in node N4. Fig. 1(b) indicates the working of Leave Copy Everywhere (LCE) where content replica is caches on each node of a requesting path. The same content is cached on three nodes N1, N2 and N3 causing redundancy in the network.

Fig. 1(c) represents the first part of LeafPopDown caching strategy. When an INTEREST packet is received from the users for the desired content at node N4, it first checks the popularity of that content in its popularity table. If this content is requested for the first time it is cached on the leaf node near the subscriber. In the given Fig. 1(c), copy of the content is cached on node N1. We can clearly see that copy of the content is cached only on single node N1 as compared to LCE that cached the content on three nodes.

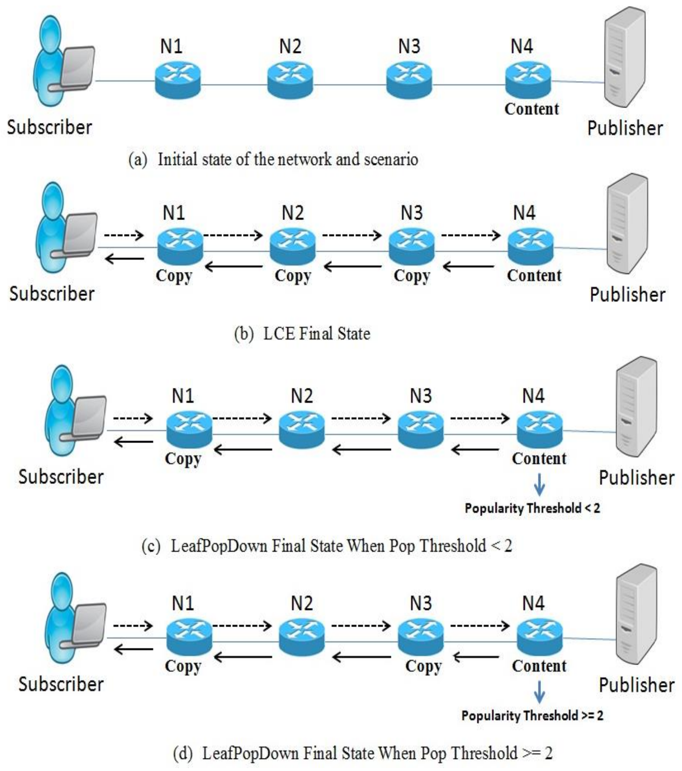
N1
N2
N3
N4
Subscriber
Content
Publisher
(a) Initial state of the network and scenario
Copy
Copy
Copy
Content
(b) LCE Final State
Copy
Content
Popularity Threshold < 2
(c) LeafPopDown Final State When Pop Threshold < 2
Copy
Copy
Content
Popularity Threshold >= 2
(d) LeafPopDown Final State When Pop Threshold >= 2

Fig. 1. LeafPopDown caching strategy.

Fig. 1(d) illustrates second part of LeafPopDown. When an INTEREST packet is received for a desired content; popularity of that content is checked if its popularity is greater than or equal to 2, it is cached on downward node of a hit node which is node N3 and on a leaf node that is N1 in the given Fig. 1(d). By comparing it with LCE, number of stored copies is less in our proposed LeafPoPDown caching policy i.e., 2 as compared to LCE that are 3. Similarly, when number of nodes increases in the request path of networks, LCE caches more copies of content causing huge redundancy in the network while LeafPopDown caching strategy cache less copies of content in the networks.

In order to conclude, we proved in the given example that our proposed caching strategy of LeafPopDown creates less redundancy as compared to LCE.

We use the following notations in algorithm of LeafPopDown caching strategy:

Symbol Notations	Description
V	The set of nodes $V = (v_1 \dots\dots v_n)$, where v_n is the number of nodes
E	Links set $E = (e_1 \dots\dots e_m)$, Here e_m is the number of links
C_i	Total contents at cache *i*
$C_{j,i}$	Content *j* at cache *i*
int^j	Interest for content *j*
r_v^j	Number of INTERESTS at node *v* for content *j*
$U(k)$	User *k*

Algorithm

```
for each ($V_n from\ i\ to\ n$)
    if
        $C_{j,i} == int^j$
            then
        if
            $r_v^j >= 2$
            then
            Cache $C_{j,i}$ at $v_{i-1}$ & $U(k+1)$
        else
            Cache $U(k+1)$
    else
        forward $int^j$ to $v_{i+1}$
```

IV. Evaluation and Analysis

Here, we discuss simulation environment and performance evaluation of LeafPopDown, LCE and MAGIC caching strategies.

A. Simulation Environment

We use SocialCCNSim [8] simulator for the evaluation of LeafPopDown, LCE and MAGIC caching strategies. This simulator is used to evaluate performance metrics of caching strategies for CCN. We conducted the simulation in chosen simulator with its inherited parameters and network topologies.

Table I shows the configured parameters for our simulations. The popularity of files has been formed following MZipf distribution in SocialCCNSim. For simulation and evaluation of LeafPopDown with LCE and MAGIC, we have selected Abilene and Tiger topologies. We set the cache size of 1 GB and catalog size to 10^6. The simulations are conducted for 86400 s. We have chosen the LRU replacement policy in the simulation. The facebook is used as a social graph for simulation. SONETOR is used as a network traffic generator.

TABLE I. Simulation Parameters

Parameters	
Popularity Model	MZipf (α = 0.88, 1.1, 1.5, 2.0)
Cache Size	1 GB
Catalog Size	10^6
Topologies	Abilene, Tiger
Repacement Policy	LRU
Traffic	SONETOR

B. Performance Evaluation

In order to have fair evaluation result, we have simulated LeafPopDown, LCE and MAGIC caching strategies in the same simulation environments for time period of one day. For evaluating performance metrics of cache hits, diversity and eviction operations, these caching strategies are simulated on two topologies of Abilene and Tiger topologies. We have taken MZipf (α = 0.88, 1.1, 1.5 and 2.0).

To summarize simulation parameters, we have taken two topologies of Abilene and Tiger, cache size of 1 GB, popularity distribution values (α = 0.88, 1.1, 1.5 and 2.0).

The simulation results of cache hits of LeafPopDown, LCE and MAGIC caching strategies are shown in Fig. 2 and 3, respectively. The results of diversity of these caching strategies are shown in Fig. 4 and 5 while results of eviction operations of these three caching strategies are shown in Fig. 6 and 7.

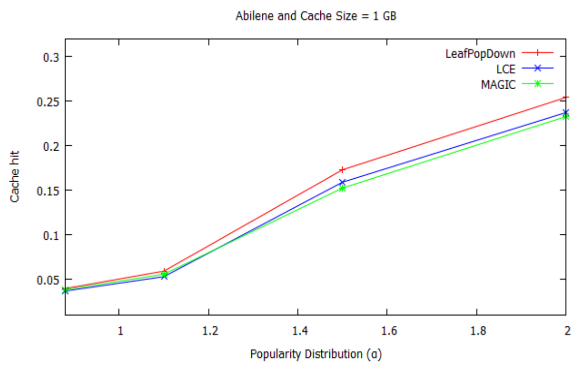

Fig. 2. Cache hits on Abilene Topology.

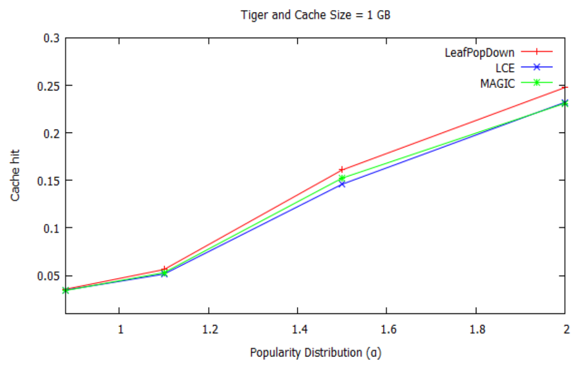

Fig. 3. Cache hits on Tiger Topology.

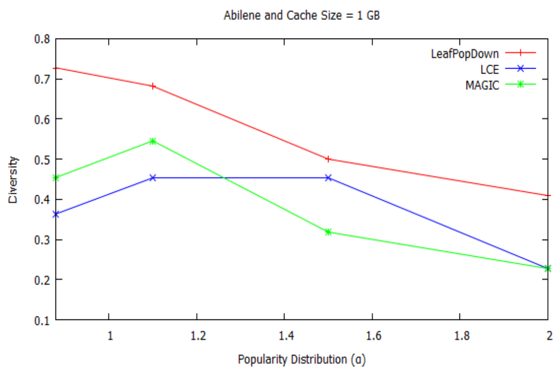

Fig. 4. Diversity on Abilene Topology.

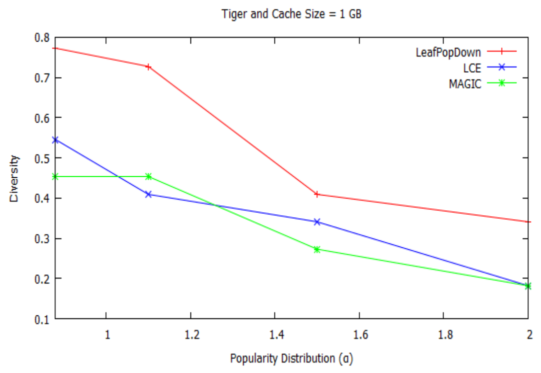

Fig. 5. Diversity on Tiger Topology.

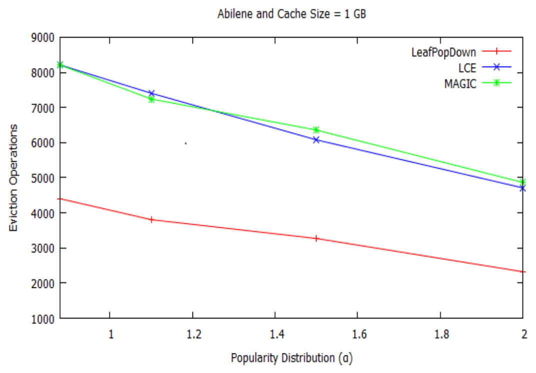

Fig. 6. Eviction operations on Abilene Topology.

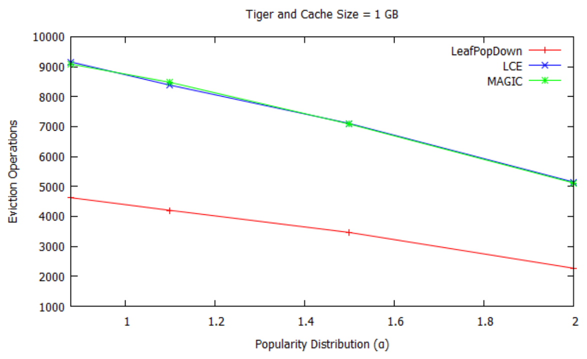

Fig. 7. Eviction operations on Tiger Topology.

By comparing the proposed caching strategy of LeafPopDown with LCE, MAGIC, we can conclude that LeafPopDown receives more cache hits as compared to other two and diversity increase significantly on our designed caching strategy. Moreover, LeafPopDown decreases eviction operations significantly as compared to LCE and MAGIC caching strategies.

V. Conclusion

In this paper, we have proposed a LeafPopDown caching strategy for ICN. LeafPopDown caches content on the leaf node near the user when it is requested in the networks. When its popularity increases to 2 or more in the popularity table and if it is requested again it leave copy of content on downward node of hit node and on leaf node near user. The simulations results show that LeafPopDown performs better than LCE and MAGIC caching policies in terms of cache hits, diversity and eviction operations. Our proposed caching strategy decrease redundancy and eviction operations while enhance the cache hits.

References

[1] Cisco, Visual networking index: Forecast and methodology, 2016-2021, Jun. 2017, White Paper

[2] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, “A Survey of Information-Centric Networking Research,” *IEEE Communications Surveys & Tutorials*, vol. 16, Iss. 12, pp. 1024-1049.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” ACM CoNEXT 2009.

[4] “SAIL NetInf,” http://www.netinf.org.

[5] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, “Developing Information Networking Further: From PSIRP to PURSUIT,” Oct. 2010.

[6] M. Zhang, H. Luo, and H. Zhang, “A survey of caching mechanisms in Information-Centric Networking,” IEEE Communications Surveys & Tutorials, vol. 17, no. 3, pp. 1473 – 1499, 2015.

[7] V. Jacobson *et al.*, “Networking named content,” in *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Rome, Italy, Dec. 2009, pp. 1–12.

[8] C. Bernardini, T. Silverston, and O. Festor, “MPC: Popularity based caching strategy for content centric networks,” in IEEE ICC, Jun. 2013, pp.3619–3623.

[9] J. M. Wang and B. Bensaou, “Progressive caching in CCN,” in *Proc. 31st IEEE Glob. Commun. Conf. (GLOBECOM)*, Anaheim, CA, USA, Dec. 2012, pp. 2727–2732.

[10] E.J. Rosensweig and J. Kurose, “Breadcrumbs: efficient, best-effort contnet location in cache networks” IEEE INFOCOM, 2009, pp. 2631-2635.

[11] K, Lee M, Park K, Kwon T. T, Choi Y, Pack S, “WAVE: Popularity-based and Collaborative In-network caching for content-oriented networks”, In Proc. *IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Orlando, FL, USA, Mar. 2012, pp. 316-321.

[12] Badov M, Seetharam A, Kurose J, Firoiu V, Nanda S, “Congestion-aware caching and search in Information-Centric Networks”, *in Proc. 1st ACM Int. Conf. Inf. Centric Netw. (ICN), Paris, France*, 2014; 37–46.