MAGIC: a Distributed MAx-Gain In-network Caching Strategy in Information-Centric Networks

Jing Ren[1], Wen Qi[2], Cedric Westphal[3,4], Jianping Wang[2], Kejie Lu[5], Shucheng Liu[3] and Sheng Wang[1]
[1] School of Communication and Information Engineering, University of Electronic Science and Technology of China
[2] Department of Computer Science, City University of Hong Kong
[3] Huawei Technologies
[4] Department of Computer Engineering, University of California, Santa Cruz
[5] Department of Electrical and Computer Engineering, University of Puerto Rico at Mayagüez

***Abstract*—Information centric networks (ICNs) allow content objects to be cached within the network, so as to provide efficient data delivery. Existing works on in-network caches mainly focus on minimizing the redundancy of caches to improve the cache hit ratio, which may not lead to significant bandwidth saving. On the other hand, it could result in too frequent caching operations, *i.e.*, cache placement and replacement, causing more power consumption at nodes, which shall be avoided in energy-limited data delivery environments, *e.g.*, wireless networks. In this paper, we propose a distributed caching strategy along the data delivery path, called MAGIC (MAx-Gain In-network Caching). MAGIC aims to reduce bandwidth consumption by jointly considering the content popularity and hop reduction. We also take the cache replacement penalty into account when making cache placement decisions to reduce the number of caching operations. We compare our caching strategy with several state-of-art caching strategies in ICNs. Our results show that the MAGIC strategy can reduce up to 34.50% bandwidth consumption, save up to 17.91% server hit ratio, and reduce up to 38.84% caching operations compared with the existing best caching strategy when cache size is small, which is a significant improvement in wireless networks with limited cache size at each wireless node.**

I. Introduction

Information centric network (ICN) aims to provide efficient content delivery based on the content name, which has been introduced in many data delivery environments, *e.g.*, wireless mesh networks (WMNs) [1]–[4]. One of the most important features of ICN is the distributed in-network caches, which has been considered as an integral part of ICNs [5]. Nodes equipped with caches can store copies of traversed content objects and satisfy subsequent content requests with the cached copies, which will greatly reduce bandwidth consumption and server load.

In many data delivery environments, especially in wireless networks, because of the limited link capacity and battery energy, it becomes quite challenging to achieve high bandwidth consumption saving through caching and avoid excessive energy consumption caused by frequent caching operations [2]–[4].

Existing works on in-network caches mainly focus on minimizing the redundancy of caches to improve the cache hit ratio [6]–[14]. Limited attention has been paid to the overall bandwidth consumption saving and caching operation reduction (in terms of the number of I/O operations to cache memory/storage). Though the content popularity has been considered in some existing caching strategies, it can only partly reflect the overall bandwidth consumption. The penalty of cache replacement is also hardly considered, which may lead to frequent caching operations.

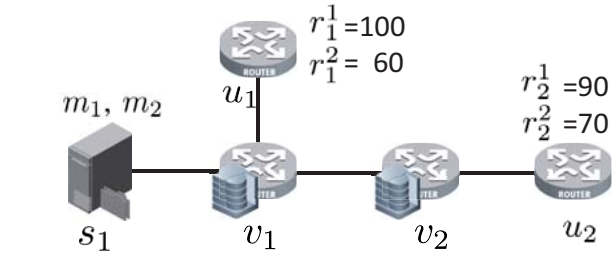


Fig. 1. An example of different caching strategies.

We use an example shown in Fig. 1 to illustrate the impact of considering bandwidth consumption saving and caching operation reduction. In this figure, a network consists of one server (s_1), two routers (v_1 and v_2) and two users (u_1 and u_2). Server s_1 holds two pieces of content (m_1 and m_2). Routers v_1 and v_2 have one unit cache capacity respectively. The user u_1 connects to router v_1 and user u_2 connects to router v_2. The requests for m_1 and m_2 from user u_2 are 90 requests/s and 70 requests/s respectively. And the requests for m_1 and m_2 from user u_1 are 100 requests/s and 60 requests/s respectively.

Let us first take a look at the existing caching strategies which make the caching placement decisions without considering the replacement penalty. In this simple example, we consider the caching decisions of ProbCache [13] which achieves better performance in terms of the number of caching operations than other caching strategies in the literature. With ProbCache, the router caches the received content objects with the same probability which is calculated based on the cache capability of the remaining routers along the path and the hop reduction from this router to the content source (original server or cache). If the router decides to cache the content and the cache is full,

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

the least recently used (LRU) content will be replaced.

Fig. 1 shows the actual cache hit ratio of each content at each router in the steady state of ProbCache. We can see that 65.44% requests for m_1 and 34.56% requests for m_2 can be satisfied by router v_1. 61.88% requests for m_1 and 38.12% requests for m_2 are responded by router v_2. This implies that frequent caching operations occur at router v_1 and v_2. The numbers of caching operations at router v_1 and v_2 are 6.06/s and 5.33/s respectively. However, if we consider the replacement penalty, some unnecessary cache replacement operations will not happen. For example, replacing m_1 at router v_2 makes $(1 - 61.88\%) \times 90 = 34.31$ requests/s for m_1 be forwarded to the server and only satisfies $38.12\% \times 70 = 26.68$ requests/s for m_2 at router v_2, which even causes worse performance in terms of both bandwidth saving and the number of caching operations. Thus, a better alternative solution is to cache m_1 at v_1 and m_2 at v_2 respectively with probability one, where no cache replacement operation will occur.

Now let us take a look at the existing caching strategies which do not take the overall bandwidth consumption saving into account. In this example, we consider one simple but efficient caching strategy which caches content based on content popularity (*i.e., the number of received requests for different content items*) [10]. Router v_2 will cache content m_1 because the number of received requests for m_1 ($r_2^1 = 90$) is larger than that for m_2 ($r_2^2 = 70$) and router v_1 will cache content m_2 as the number of received requests for m_2 ($r_2^2 + r_1^2 = 130$) is larger than that for m_1 ($r_1^1 = 100$ as v_2 has satisfied 90 requests originated from u_2 for m_1). The total bandwidth consumption is $r_2^1 + r_2^2 \times 2 + r_1^1 \times 2 + r_1^2 = 490$. However, if we cache m_1 at router v_1 and m_2 at router v_2, the total bandwidth consumption is only $r_2^1 \times 2 + r_2^2 + r_1^1 + r_1^2 \times 2 = 470$. From this simple example, we can see that both content popularity and hop reduction should be taken into account in order to save the overall bandwidth consumption.

Based on the above observations, we propose a distributed caching strategy along the data delivery path, called MAGIC (MAx-Gain In-network Caching), to improve the bandwidth consumption saving by jointly considering the content popularity as well as hop reduction and to reduce the number of caching operations by taking the cache replacement penalty into account when making cache placement decisions.

The main contributions of this paper are summarized as follows.

- We propose a new notion, referred to as cache gain, to facilitate the caching decision. The cache gain jointly considers content popularity as well as hop reduction which reflects the actual bandwidth consumption saving. It also takes into account the cache replacement penalty to minimize the number of caching operations.
- We design a distributed caching strategy to cooperate with in-network caches along the delivery path. We extend the request (*i.e., INTEREST* message) to find the maximal local cache gain along the path, called *MaxGain*. Then the returned content (*i.e., DATA* message) will be cached at the node, the local cache gain of which is equal to the maximal local cache gain.
- We compare our MAGIC caching strategy with several well-known caching strategies and our results show that the strategy proposed by us could achieve up to 34.50% improvement in bandwidth consumption saving, 17.91% improvement in the reduced server hit ratio, and up to 38.84% improvement in the number of reduced caching operations against the best performance cache placement and replacement strategy in the literature.

The remainder of this paper is organized as follows. In Section II, we discuss the related work on caching strategies. Section III describes the MAGIC strategy. Section IV presents the evaluation results and Section V concludes this paper.

II. Background and Related work

Caching placement strategies have been widely investigated for web caching [15], [16]. However, the caching placement problem in ICN differs from that in web proxies because the cache capacity of routers in ICN is quite limited. Although using techniques like web caching may facilitate the incremental deployment of ICN, [17] shows that the performance gap between extended pervasive caching at in-network nodes and caching at edge nodes reaches the peak when the individual cache sizes is $\approx 2\%$ of total objects. In this section, we will give a brief overview of the caching strategies, including cache placement and replacement strategies, used in ICN architectures.

Researches on cache placement can be grouped into two types: off-path and on-path cache placement strategies. The off-path cache placement strategies aim to find the global optimal caching placement based on content popularity and cache status of all caching nodes [6], [7]. However, collecting and disseminating such information among all caching nodes will cause additional overhead and may not achieve good performance [18].

The on-path cache placement strategies focus on caching certain content at the right locations along the delivery path [8]–[14]. The nodes make cache decisions based on local information [8]–[10]. Furthermore, they can exchange some limited information and infer the near-optimal cache placement [11]–[14]. However, these strategies mainly focus on minimizing the redundancy of caches to improve the cache hit ratio.

The most commonly used on-path caching strategy in ICN is *leave copy everywhere (LCE)* which caches the content objects at all on-path nodes [8]. Another simple strategy, called *random autonomous caching (RAC)*, caches the content objects at each individual node with a fixed probability p [9]. *Most frequently used placement (MFUP)* strategy counts the content popularity and caches the most popular content objects [10]. However, these caching nodes do not cooperate with each other, which results in unnecessarily redundant copies of the content cached in the network.

To cooperate with the on-path caches, *random algorithm (RDM)* extends the RAC strategy by randomly selecting one

caching node along the path to cache the content [11]. *Leave Copy Down (LCD)* [12] pushes a copy of the requested content one hop closer to the client each time when a content request is satisfied. LCD considers the content's popularity while it does not introduce additional overhead. The *Betweenness Centrality (BetwCent)* strategy leverages the concept of betweenness centrality in graph theory and caches the content at the caching node which has more number of content delivery paths passing through it compared with other on-path nodes [11]. *ProbCache* makes the cache placement decision based on the cache capability of the remaining nodes along the path and the hop reduction gains [13]. However, as the main targets of these strategies are to reduce the cache redundancy and improve the cache hit ratio, limited attention has been paid to the overall bandwidth consumption saving and caching operations reduction.

In ICN, two of the most popular cache replacement strategies (which determines what to replace) are *Least Recently Used (LRU)* and *Least Frequently Used (LFU)* [8]. LRU considers that the recently used content will be re-requested with high possibility in the period of time in the future. Thus, the content which has not been accessed for the longest time will be replaced. LFU counts the hit frequency of the cached content and replaces the content with the least hits number in the recent past. *Least Benefit (LB)* improves LFU by considering hop reduction gains [14]. LB is defined as the product of content hit ratio and hop reduction which is similar as our definition of cache gain. However, LB is used as the local replacement strategy and our strategy tries to find the node with the maximal local cache gain along the data delivery path.

III. MAGIC: MAx-GAIN IN-NETWORK CACHING

In this section, we first state the system model and then elaborate on our caching strategy.

A. System Model

In this paper, we consider an ICN with an arbitrary topology as a directed graph $G =< V, E >$, where V denotes the set of nodes, each of which has a limited cache capacity, and E is the set of links between nodes. To facilitate further discussion, we summarize the main notations to be used in this paper as follows.

- V: The set of nodes where $V = \{v_1, \cdots, v_{|V|}\}$, where $|V|$ is the number of nodes.
- E: The set of links between nodes.
- c_i: The cache capacity of node v_i in terms of the number of content units.
- M: The content requested in the network where $M = \{m_1, \cdots, m_{|M|}\}$.
- R: The set of requests.
- C_v: The content cached at node v where $C_v \subset M$.
- r_v^m: The request rate (the number of *INTEREST* messages per second) at node v for content m.
- h_v^m: The hop counts from node v to the original server which holds content m.

To ease the description, we consider the basic request forwarding follows the NDN specification [8]. The routing node forwards the request message, *i.e., INTEREST* message, along a path towards the original server unless it has a copy of the desired content. The routing table, *i.e.,* forwarding information base (FIB), is obtained through the OSPFN (OSPF for Named-data) protocol which is designed for NDN to advertise the name prefixes by extending opaque link state advertisements [19]. We assume that routing nodes will not advertise false information to pollute caches. The content data, *i.e., DATA* message, will be routed along the same path with the *INTEREST* message by using the pending interest table (PIT).

For simplicity, we assume the cached content units have the same size [11], [12], [14]. We also consider that each routing node will count the request rate for each content requested in the recent period to get the content popularity over a fixed time period T [10]. Each caching node can obtain the number of hops from it to the original server, *i.e., hop reduction*, based on the information exchanged by the OSPFN protocol.

B. MAGIC caching strategy

Based on the observations from the simple example shown in Fig. 1, we aim to provide a caching strategy to reduce the overall bandwidth consumption and the number of caching operations from the on-path caching perspective.

To do this, we first define the potential *cache placement gain* (the gain if caching a new content at the node) and *cache replacement penalty* (the loss if evicting a cached content from the node) by jointly considering the content popularity and hop reduction which reflect the saved bandwidth consumption. We then define the potential *local cache gain* for a node which combines the potential *cache placement gain* and *cache replacement gain* to minimize the number of caching operations. Finally, we design a distributed caching strategy, MAGIC, to cooperate in-network caches along a path.

1) ***Potential Local Cache Gain****:* To minimize the number of caching operations, we consider that node v should consider the penalty of replacing a cached content from the cache memory/storage when making the cache decision for the new requested content m. For this purpose, we should first obtain the potential *cache placement gain* and *cache replacement penalty*.

To reflect the bandwidth consumption saving for caching a request content m at node v, we define the cache placement gain $PlaceGain_v^m$ as the product of historic request rate for content v (content popularity) and the number of hops from node v to the original server (hop-reduction).

$$PlaceGain_v^m = r_v^m \times h_v^m \tag{1}$$

It is worth to note that we use h_v^m to reflect the bandwidth consumption saving. However, we can also use other metrics, such as transmission delay or link weight. Note that h_v^m can also be defined as the the number of hops from node v to the node which actually returns the content. In our evaluation, we find that the results are similar for these two definitions. What is more, h_v^m can be obtained with less operations when h_v^m is defined as from node v to the original server.

The cache replacement penalty, $RePlacePenalty_v$, for node v is defined as the minimal bandwidth consumption saving of

TABLE I
PSEUDO-CODE FOR MAGIC CACHING STRATEGY

INTEREST message
1. Get $\{LocalGain_v^m\}$
2. **if** $LocalGain_v^m > MaxGain$
3. update($MaxGain$) in the header
4. **if** cache miss **and** no matched PIT entry
5. **then** creates PIT entry and records the $LocalGain_v^m$ then forward the packet to the next hop towards the original server
5. **else**
6. send back *DATA* message with the $MaxGain$
DATA message
1. find PIT entry and Get($LocalGain_v^m$)
2. **if** $LocalGain_v^m == MaxGain$
3. **then** cache the content
4. forward the packet towards client

all cached content objects.

$$RePlacePenalty_v = \begin{cases} \min_{m \in C_v} r_v^m \times h_v^m, & \text{if } |C_v| = c_i \\ 0, & \text{if } |C_v| < c_i \end{cases} \quad (2)$$

Finally, we define the potential local cache gain as the difference of the potential cache placement gain and cache replacement penalty.

$$LocalGain_v^m = PlaceGain_v^m - RePlacePenalty_v \quad (3)$$

If the gain for caching a new content is smaller than the penalty for replacing any cached content, the cached content will not be evicted and no cache replacement will occur in MAGIC.

*2) **The Distributed Caching Strategy**:* Based on the local cache information, *i.e.,* the potential $LocalGain_v^m$, a distributed caching strategy is needed to find the best place along the data delivery path for caching content m. Fig. 2 shows the basic operations of MAGIC which operates at per request level similar to ProbCache [13].

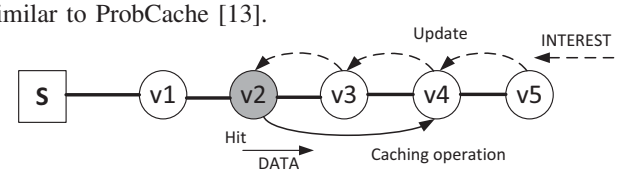


Fig. 2. MAGIC operations

To determine the node with the maximal *LocalGain* for content m, we extend the *INTEREST* message to carry the maximal local cache gain along the path in the new defined *MaxGain* field. To request content m, an *INTEREST* message is sent and the *MaxGain* value in the header is set to 0. When receiving the *INTEREST* message, each router v first calculates the $LocalGain_v^m$ and compares it with the value recorded in the *MaxGain* field. If the local cache gain of router v is larger than the recorded *MaxGain* value, router v will update the *MaxGain* value in the *INTEREST* message. When the *INTEREST* message reaches the original server or a cache, the *MaxGain* field in the *INTEREST* message is the maximum local cache gain along the delivery path.

We also extend the *DATA* message to piggyback the *MaxGain* value collected by the *INTEREST* message. Along the delivery path (the reverse path of the *INTEREST* message), the router, whose $LocalGain_v^m$ is equal to the *MaxGain* value in the *DATA* message, will do the caching operation. Table I shows the pseudo-code for forwarding both the *INTEREST* and *DATA* messages.

IV. PERFORMANCE EVALUATION

A. Evaluation Environments and Performance Metrics

1) Evaluation Environments: To evaluate our proposed caching strategy, we conduct the performance evaluation with a custom-built simulator, in which all nodes are cache enabled. As the caching strategies to be compared in this paper are not integrated into the well-known simulation framework, e.g., ccnSim, we use the custom-built simulator for the convenience of implementation. The specification of all these existing caching strategies to be compared in this paper is strictly followed. We compare the performance of our strategy against all on-path caching strategies mentioned in the related work. However we only illustrate the results of the following four different strategies due to the space limitations:

- **MFUP**, which caches c_i most frequently used content units [10].
- **LCE**, which caches content at all the intermediate nodes [8].
- **ProbCache**, which cooperatively caches content on a path [13].
- **RAC**, which caches content at each node with a fixed probability $p = 0.5$ [9].

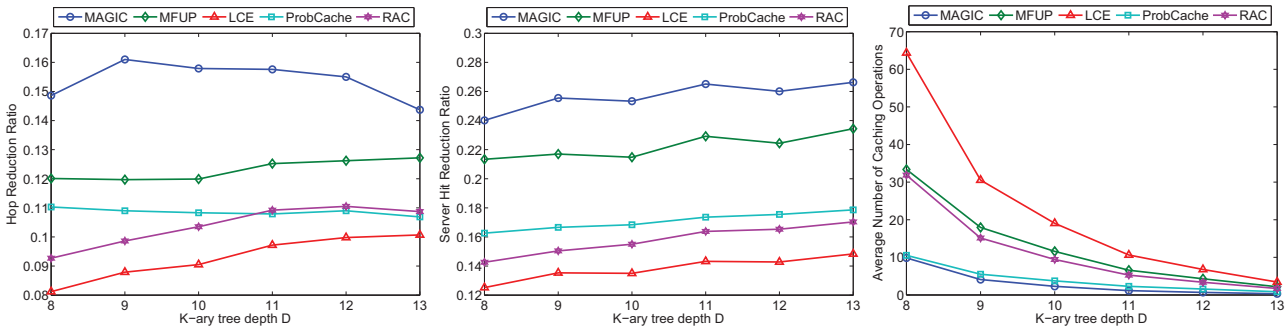
For ProbCache and RAC, we use LRU as the cache replacement algorithm. For our MAGIC and MFUP, each caching node counts the content popularity as the number of received requests per second.

we use a non-complete K-ary tree topology and assume that the root node is the server and all the requests are sent from the leaf nodes, similar to the setting of [11]. In the evaluation, our cache-size unit and the basic content unit are a packet for simplicity. For each test round, we generate 4×10^5 requests to allow the system to reach the steady state. And the content requests follow the Zipf distribution. The number of content units in the network is 1×10^4.

In the evaluation, we consider the effect of three parameters

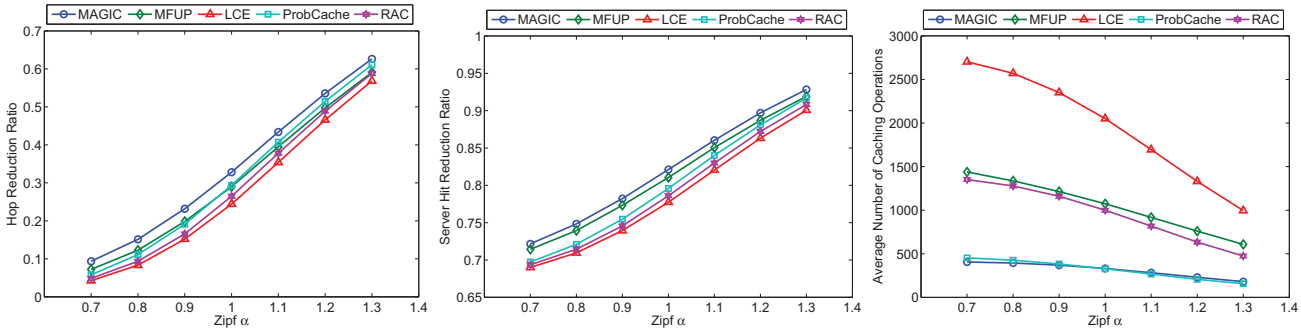
- D: the depth of the tree from the root, $D \in [8, 13]$.
- ca_{size}: the cache capacity in terms of content units, $ca_{size} \in [5, 55]$.
- α: the parameter of Zipf distribution, $\alpha \in [0.7, 1.3]$.

For K-ary tree topology, there are two parameters: k which denotes the maximum number of children of each node and D which presents the depth of the tree. As we find the effect of k is limited, we use $k = 5$ in the evaluation where we randomly generate children nodes for each node in the range of [0, 5]. In the evaluation, we assume homogeneous cache



(a) Hop reduction ratio β (b) Server hit reduction ratio γ (c) Average number of caching operations δ

Fig. 3. Algorithms' performances with different topologies. ($\alpha = 0.8$, $ca_{size} = 20$)



(a) Hop reduction ratio β (b) Server hit reduction ratio γ (c) Average number of caching operations δ

Fig. 4. Algorithms' performances with different Zipf α ($ca_{size} = 20$, $D = 8$).

size for simplicity, *i.e.*, $c_i = ca_{size}\ \forall i \in \{1, \cdots, |V|\}$. We set ca_{size} with a range of 5 to 55 content units.

2) Performance Metrics: We evaluate the performance of the caching strategies from three aspects:

- **Hop reduction ratio**, which indicates the saved bandwidth consumption.

$$\beta = \frac{\sum_R h}{\sum_R H} \tag{4}$$

  where h is the hop counts from the node (where a cache hit occurs) to the original server and H is the hop counts from the client to the original server for a content. Note that we can also use other metrics, such as transmission delay or link weight, to reflect the saved bandwidth.
- **Server hit reduction ratio**, which reflects the load saving of servers due to the in-network cache hits.

$$\gamma = \frac{w}{|R|} \tag{5}$$

  where w is the number of requests satisfied by in-network caches and $|R|$ is the total number of requests.
- **Average number of caching operations**, which defined as the average number of I/O operations of cache memory/storage at all nodes.

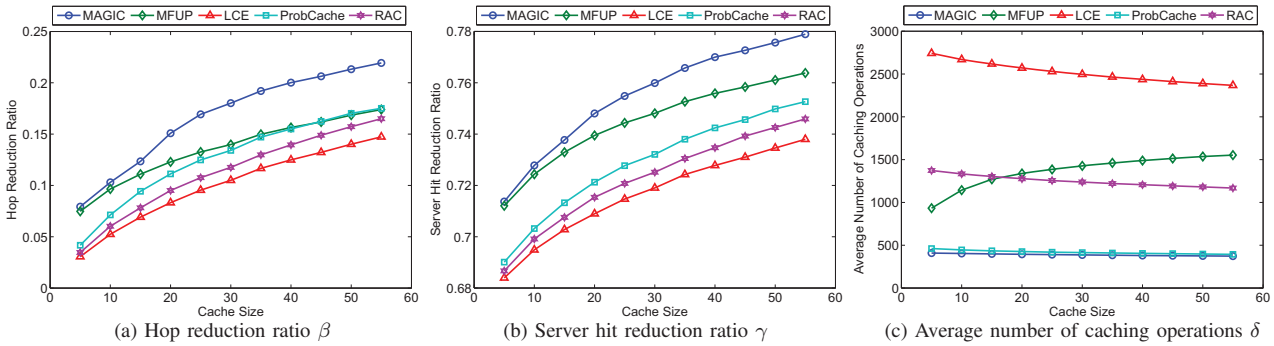
$$\delta = \frac{\Sigma_V o_i}{|V|} \tag{6}$$

  where o_i is the number of caching operations at node v_i.

B. Performance Results

1) The depth of K-ary tree, D: Fig. 3 shows the impact of depth D of K-ary tree on the aforementioned three performance metrics. Regarding to hop reduction ratio and server hit reduction ratio, MAGIC attains its maximum performance when topology depth is 9. The maximal improvements of MAGIC over MFUP in terms of bandwidth consumption saving and server hit reduction ratio are $\frac{0.161-0.1197}{0.1197} = 34.50\%$ and $\frac{0.2533-0.2148}{0.2148} = 17.92\%$, respectively However, the number of caching operations of all compared caching strategies decreases with the increase of topology size because the overall cache capacity increases and less cache replacement will occur. The maximum performance improvement of MAGIC against the second best performance strategy, the ProbCache, reaches $\frac{-(2.2878-3.7405)}{3.7405} = 38.84\%$ when the topology depth is 10.

2) The parameter of Zipf distribution, α: Fig. 4 shows the impact of different Zipf α on the three performance metrics. In Fig. 4, we fix $D = 8$ and $ca_{size} = 20$. From Fig. 4, we can see that both the hop reduction ratio and server hit ratio of all these caching strategies increase with the increase of α, while the number of caching operations decreases as the increase of α. When α of Zipf distribution increases, the popularity of the hot content increases. Thus, the benefits of caching these most popular content will also increase and less cache replacement will occur. When $\alpha < 1.0$, MAGIC has the least number of caching operations (compared to ProbCache, the advantage is up to 10.48%). When $\alpha > 1.0$, in terms of the number of caching operations, MAGIC performs sightly



(a) Hop reduction ratio β (b) Server hit reduction ratio γ (c) Average number of caching operations δ

Fig. 5. Algorithms' performances with different cache sizes ($\alpha = 0.8$, $D = 8$).

worse than ProbCache. However, MAGIC always outperforms ProbCache in terms of hop reduction ratio and server hit reduction ratio. Regarding to the server hit ratio, MAGIC can only achieve up to 1.31% improvement when α = 0.8 against MFUP. However, from the figure, we can see that the hop reduction advantage of MAGIC is around 10-20% (presenting the bandwidth consumption saving) against MFUP. Especially, when the Zipf is smaller (*i.e.,* α = 0.7), MAGIC can achieve up to 29.61% improvement compared with MFUP.

3) The size of cache, ca_{size}: Figure 5 shows the evaluation results when the cache size increases. As the caching operations, e.g., searching for a particular content in a cache with large capacity, will cause delay, we expect that the cache capacity of ICN routers in the future will not be very large as the routers need to forward packets in line-speed. Regarding to the hop reduction ratio and server hit reduction ratio, the performance improvement of MAGIC over MFUP becomes larger as the increase of cache size when $ca_{size} <= 30$ (the maximum advantages are around 28.84% and 1.98%, respectively), and remains stable when $ca_{size} > 30$. For the number of caching operations, MAGIC still performs the best among all the strategies. The maximal improvement of MAGIC over the second best strategy, the ProbCache, is 11.32%.

V. Conclusion

In this paper, we have explored the caching strategy of ICNs to reduce the network bandwidth consumption and caching operations in terms of I/O operations of cache memory/storage, considering the limited link capacity, cache size, and energy at each node in the wireless networks. We have proposed a distributed caching strategy along a data delivery path, called MAGIC (MAx-Gain In-network Caching), which aims to reduce bandwidth consumption by jointly considering the content popularity and hop reduction. We also take the cache replacement penalty into account when making cache placement decisions to reduce the number of caching operations. We compare our algorithm with several state-of-art caching strategies in ICNs. And the evaluation shows that our strategy performs well when the cache capacity is limited. In a wireless network with lossy transmission, the performance of caching strategies may be affected. In our future work, we will implement the proposed caching strategy into some existing simulation frameworks which have simulated the communications in wireless networks.

References

[1] K. Pentikousis and B. Ohlman, "ICN baseline scenarios," Internet Draft, draft-pentikousis-icn-scenarios (work in progress), Tech. Rep., 2013.

[2] J. Wang, K. Lu, S. Zhang, J. Fan, Y. Zhu, and B. Cheng, "An efficient communication relay placement algorithm for content-centric wireless mesh networks," *International Journal of Communication Systems*, 2013.

[3] M. Amadeo, A. Molinaro, and G. Ruggeri, "An energy-efficient content-centric approach in mesh networking," in *Proc. of IEEE ICC*, 2012, pp. 5736–5740.

[4] B. Azimdoost, C. Westphal, and H. Sadjadpour, "On the throughput capacity of information-centric networks," in *Proc. of ITC*, 2013, pp. 1–9.

[5] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[6] V. Sourlas, P. Flegkas, G. S. Paschos, D. Katsaros, and L. Tassiulas, "Storage planning and replica assignment in content-centric publish/subscribe networks," *Computer Networks*, vol. 55, no. 18, 2011.

[7] SAIL, "NetInf content delivery and operations," SAIL project, SAIL Project Deliverable D-3.2, 2011.

[8] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of ACM CoNEXT*, New York, NY, USA, 2009, pp. 1–12.

[9] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *Proc. of ReARCH Workshop*, New York, NY, USA, 2010, pp. 5:1–5:6.

[10] M. Korupolu and M. Dahlin, "Coordinated placement and replacement for large-scale distributed caches," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 6, pp. 1317–1329, 2002.

[11] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache Less for more in information-centric networks," in *NETWORKING 2012*, ser. Lecture Notes in Computer Science, 2012, no. 7289, pp. 27–40.

[12] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.

[13] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. of ICN Workshop on Information-centric Networking*, New York, NY, USA, 2012, pp. 55–60.

[14] S. Wang, J. Bi, J. Wu, Z. Li, W. Zhang, and X. Yang, "Could in-network caching benefit information-centric networking?" in *Proc. of AINTEC*, New York, NY, USA, 2011, pp. 112–115.

[15] S. Bhattacharjee, K. Calvert, and E. Zegura, "Self-organizing wide-area network caches," in *Proc. of IEEE INFOCOM*, 1998, pp. 600–608.

[16] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, "On the scale and performance of cooperative web proxy caching," in *Proc. of ACM SOSP*, New York, NY, USA, 1999, pp. 16–31.

[17] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: incrementally deployable ICN," in *Proc. of ACM SIGCOMM*, New York, NY, USA, 2013, pp. 147–158.

[18] Y. Wang, K. Lee, B. Venkataraman, R. Shamanna, I. Rhee, and S. Yang, "Advertising cached contents in the control plane: Necessity and feasibility," in *Proc. of IEEE INFOCOM WKSHPS*, 2012, pp. 286–291.

[19] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: an OSPF based routing protocol for named data networking," *University of Memphis and University of Arizona, Tech. Rep*, 2012.