A Comparison of Caching Strategies for Content Centric Networking

César Bernardini, Thomas Silverston, Olivier Festor

A Comparison of Caching Strategies for Content Centric Networking

César Bernardini* Thomas Silverston† and Olivier Festor‡§
*University of Trento, Italy
†University of Tokyo, JFLI CNRS UMI 3527, Tokyo, Japan
‡Université de Lorraine, LORIA CNRS UMR 7503, Vandoeuvre-les-Nancy, France
§Inria Nancy – Grand Est, Villers-les-Nancy, France

***Abstract*—Content Centric Networking (CCN) is a new architecture for a future Internet. CCN relies on in-network caching capabilities of nodes and the efficiency of this architecture depends drastically on performances of caching strategies. Thus, there have been a lot of studies proposing new caching strategies to improve the performances of CCN. However, among all these strategies, it is still unclear which one performs better as there is a lack of common environment to compare these strategies.**

In this paper, we compare the performances of CCN caching strategies within the same simulation environment. We build a common evaluation scenario and we compare via simulation five relevant caching strategies: Leave Copy Everywhere (LCE), Leave Copy Down (LCD), ProbCache, Cache "Less" For More and MAGIC. We analyze the performances of all the strategies in terms of Cache Hit, Stretch, Diversity and Complexity, and determine the cache strategy that fits the best with every scenario.

I. Introduction

Information Centric Networking (ICN) is a new network paradigm for a Future Internet [1]. It is based on *name-based* communication model instead of the traditional *host-based* communication model of the Internet and includes other features such as in-network caching, routing by name, multicast and encryption support. To date, there have been several ICN proposals such as Content Centric Networks (CCN) [2], PURSUIT [3] and NetInf [4]. Among them, the CCN architecture and its real implementation into CCNx [5] have received the most of the attention from the research community.

CCN relies on in-network caching and multiple copies of the data are stored in the network. Any nodes having the data can serve future requests, and it helps reducing the load on servers and congestion in the network. Thus, many research works have proposed novel caching strategies in order to improve the performances of CCN, starting from the CCN default strategy *Leave Copy Everywhere* (LCE) [6], to more sophisticated strategies such as *Leave Copy Down* (LCD) [6], *ProbCache* [7], *Cache "Less" for More* [8] and *MAGIC* [9].

However, these caching strategies are usually evaluated within different simulation environments, using a wide variety of parameters such as different topologies, catalogs, workload traces, content popularity models, cache sizes and so on. Furthermore, these proposed caching strategies have never been compared with each other, making it almost impossible to draw any conclusions from these works. In other words, it is still impossible to answer simple questions about caching strategies such as "which one really improves the overall performances of CCN in realistic scenarios", or "which strategy performs better than others with respects to the same comparison framework".

To this end, in this paper, we evaluate and compare the most relevant CCN caching strategies with a common evaluation scenario, using the same simulation environment, parameters and metrics. The contributions of this paper are the following. We first analyze the simulations scenarios commonly used in the literature and discuss the diversity of the parameters. We propose a common evaluation framework handling the most relevant parameters that are used in the evaluation process. We implemented and evaluate under our simulation framework the five most relevant caching strategies and compare them with respects to common metrics.

The remainder of the paper is organized as follows. We first survey the related work in Section II and emphasize on the different simulation environments and parameters used to evaluate the caching strategies. Then, we propose in Section III a common evaluation scenario for comparing the caching strategies. The Section IV describes the simulation environment of the presented evaluation scenario. The Section V shows the results of our comparison study and discusses the complexity of each strategy. Finally, we summarize our findings in Section VI, conclude and present future work in Section VII.

II. Related Work

Since CCN has been proposed by Jacobson et al. in 2009 [2], this research topic has become very active. Especially, a lot of works focus on the design of new caching strategies, as it is a key point of this novel networking architecture [6–9]. However, these strategies are usually evaluated with dedicated simulation tools, under different simulation environments, scenarios and input parameters. Thus, the lack of common evaluation framework makes it impossible to compare different caching strategies between them.

From the literature, we extracted four parameters to emphasize on the diverse simulation environments: topology, content popularity model, content catalog and cache size.

Topology: In the literature, the topologies used to evaluate the CCN caching strategies vary from k-ary trees to complex

ISP level topologies or a combination of all of them: [7] uses a 6-level binary tree (i.e. 127 nodes) where all the requests are issued from the last two levels of the ISP and are served by the root node. [8] and [9] use the same kind of a k-ary tree ranging from 4 to 6 levels and the number of children per node varies between 2 and 5. [10] uses a mixed approach, where an ISP level topology is generated and a binary tree of depth 5 is built at each node of the ISP level topology.

Rossi et al. [11] [12] uses the educational network Abilene, and other ISP level topologies such as GEANT, DTelecom, Level3 and Tiger. [13] considers a 4x4 torus topology and ISP level topology (GEANT). [14] serves of RocketFuel to generate ISP level topologies in particular AS1755 (eBone) and AS3967 (Exodus) and serves of BRITE to generate two other hierarchical topologies. [13] and [15] only consider the GEANT topology.

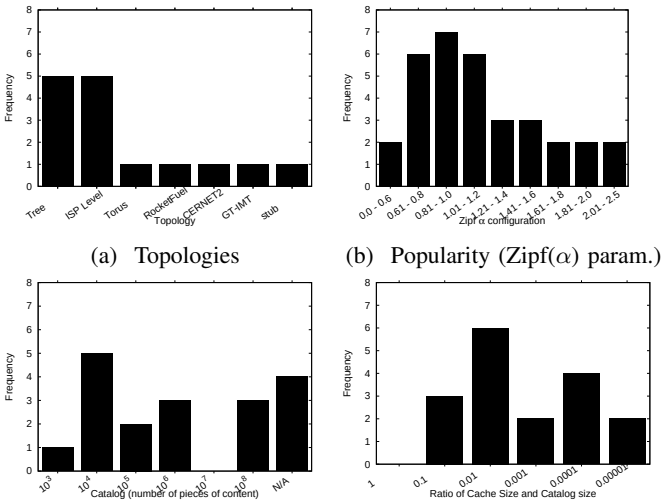
Content Popularity Model: The content popularity model is a function that establishes the popularity of every piece of content, i.e., how often every single piece of content is going to be requested. The content popularity is commonly modeled with a probability distribution function such as a Zipf or MZipf [16] [11]. In the literature, the (M)Zipf α parameter ranges largely from 0.6 to 2.5. For instance, the catalog of the PirateBay is modeled with $\alpha = 0.75$, DailyMotion with $\alpha = 0.88$, while the VoD popularity in China exhibits a α parameter ranging from 0.65 to 1.0 [17]. [7] uses a Zipf popularity model with α of 0.8. In [8], this parameter is 1.0 while [15] varies it between 0.6 and 1.1. In [13] [11], α ranges between 0.65 and 2.5. The α varies between 0.7 and 0.96 in [14] while [9] evaluates with an α between 0.7 and 1.1.

Other authors have resorted to real traces: [10] evaluates the caches with traces extracted from Akamai CDN Asia; [18] resorts to traces from wdklife.com, a Chinese entertainment feed.

Catalog: The catalog represents the entire collection of elements (i.e.: content) in the network. The number of requests for a certain piece of content depends therefore directly on the content popularity model. One more time, the related work uses a wide range of values for the catalog size: on the one hand, [13] [11] [12] consider a Youtube-like catalog with 10^8 pieces (10 MB each), while on the other hand [9] uses a catalog size of only 10^4 pieces. Differently, [18] uses 8×10^4 pieces of content for its catalog and [8] uses only 10^3. Surprisingly, [7], [10], [15], [14] do not mention the size of the catalog; [7] [10] [15] only detail that they generated 10^5 requests for all the catalog.

Cache Size: The cache size determines the space available in every CCN node for storing temporally pieces of content. This size is usually expressed with an absolute value or a ratio with regards to the catalog size (i.e. 10 elements or 0.01 of the catalog size respectively). In order to compare caching strategies together, it makes more sense to use a ratio for the cache size as most of quoted works use different catalog size.

Again, there is a wide different size of cache in the literature, with size ranging from 2×10^2 to 10^5 elements. [15] experiments values of 0.2, and [8] a value of 0.1. In [14], it ranges between 0.01 and 0.06 while [9] tests values from 0.0055 to 0.005. [11] uses the larger difference between cache size and catalog with ratio of 10^5.



(a) Topologies (b) Popularity (Zipf(α) param.)

(c) Catalog (d) Cache Size

Fig. 1: Range of parameter values in different simulation environments. These values are extracted from the surveyed papers in the related work (Section II).

III. Common Evaluation Scenario

As stated before, caching strategies for CCN are evaluated through very different simulation environments. In Figure 1, we plotted the histograms of the four parameters of all the papers surveyed in the previous section: topology, popularity (α parameter), catalog and cache size. From this figure, it is clear that there is no consensus for any specific parameter values; we observe instead a vast range of different values for all these parameters.

A. Parameters

In this paper, we define a common evaluation scenario that relies on the most common parameters used across the literature in order to compare all the caching strategies.

Regarding the topology, we consider a Future content-centric Internet built on the CCN architecture. This new Internet will still be organized with ISPs and the topological structure of this Future Internet will be just like today's Internet. ISP topologies are therefore the best candidates for evaluating strategies compared with trees or torus topologies. We choose four ISP level topologies: Abilene, Dtelecom, GEANT and Tiger [11].

With regard to the content popularity model, we serve of a popularity model based on Zipf probability distribution. The α parameter of the Zipf varies between 0.65, 1.1, 1.5 and 2.0. The 0.65 value refers to a low popularity scenario when the probability of selecting a piece of content is close to a uniform probability function. The 1.1 value stands for a normal popularity scenario. Finally, we consider two high popularity scenarios with α equals to 1.5 and 2.0. In the rest of this paper,

we refer to the scenario as *low*, *normal* and *high popularity* scenario.

As catalog size, from Fig. 1c, 10^4 elements was the more frequently used value. However, this is a very limited catalog size. We thus consider in our experiments a catalog of 10^6 elements, as it is also a common value. The elements of the catalog are randomly distributed across all the nodes of the network.

The cache size will vary from 1 to 1,000 elements, which corresponds to 10^{-6} to 10^{-3} with respect to the catalog size, and we call it Cache size ratio. We consider homogeneous cache size (all nodes have the same cache size). The cache replacement policy used in the comparisons is Last Recently Used (LRU) as it is shown in [19] that cache replacement policies may be grouped into the same equivalent class, i.e., different strategies will exhibit the same performances.

Parameters	
Catalog	10^6
Popularity Model	MZipf($\alpha = \{0.65; 1.1; 1.5; 2.0\}, \beta = 0$)
Topologies	ISP (Geant, Dtelecom, Abilene, Tiger)
Cache size	$\{10^{-6}; 10^{-5}; 10^{-4}; 10^{-3}\}$

TABLE I: Common evaluation scenario parameters

We summarize all the chosen parameters of the scenario in the Table I.

B. Caching Strategies

We describe the main caching strategies proposed in the literature that we will compare with our general framework.

- *Leave Copy Everywhere* (LCE) [2] is the default CCN strategy: every CCN node stores a copy of every *Data* message that has passed by it;
- *Leave Copy Down* (LCD) [6]: When a cache hit occurs, this caching strategy copies the element at the direct neighbor, from which the requests comes from;
- *MAx-Gain In-Network Caching* (MAGIC) [9]: MAGIC aims at maximizing the local cache gain of inserting a new piece of content and minimizing the cache replacement penalty of removing an old piece of content;
- *Probabilistic in-network Caching* (ProbCache) [7] is a probabilistic algorithm for distributed content caching along a path of caches. ProbCache selects the best node to cache the content in a path of caches. It pretends to leave caching space for other flows and fairly distribute content through a shared path;
- *Cache "Less For More"* [8]: This strategy aims at caching at the best location in a path of caches by using the concept of *betweenness centrality*.

IV. Simulation Environment

In order to compare caching strategies for CCN under the same simulation environment and common evaluation scenario, we implemented a discrete-event simulation tool written in Python, available at [20]. This simulation tool needed to be able to get all the parameters and scenarios we have presented throughout the paper. Then we implemented the five previously mentioned caching strategies in our simulator.

Simulations	
#Runs	3
Simulated Time	1 day
Evaluation Metrics	Cache Hit, Stretch Diversity, Complexity
CCN Cache Configuration	
Replacement Policy	LRU
Caching Strategies	{LCE, LCD, ProbCache, Cache "Less" For More, MAGIC}

TABLE II: Simulation Environment

We then perform simulation experiments for the five caching strategies by varying the parameters of the common evaluation scenario (Table I). For each simulation experiment, three runs are performed and we present confidence interval. Before running the simulation, the requests are placed randomly in the nodes of the topology.

Next section presents the metrics to evaluate the performances of the caching strategies. We summarize the simulation environment and the evaluation metrics in the Table II.

A. Evaluation Metrics

Along this paper, caching strategies are compared with the following metrics: Cache Hit, Stretch, Diversity and Computational complexity.

When we analyze an individual cache, we report a Hit operation if an element is found in a cache otherwise a Miss operation. In a network of caches as CCN, we measure efficiency of caches with Cache Hit shown in Equation 1: $hits_i$ refers to the number of *Interest* messages answered by the cache of node i, while $miss_i$ the number of unanswered *Interest* messages. $|N|$ refers to the number of nodes in the topology.

$$CacheHit = \frac{\sum_{i=1}^{|N|} hits_i}{(\sum_{i=1}^{|N|} hits_i) + \sum_{i=1}^{|N|} miss_i} \quad (1)$$

The Stretch is also a very common metric to evaluate caching strategies [14] [7] [8] [12] [11] [9]. Stretch defines the percentage of the path that has been travelled to retrieve the content. Giving that $hops\_walked_i$ is the number of hops from the request i to the node caching the content and, $total\_hops_i$ is the hop count from user (request i) to the original server, Stretch is the ratio of the complete path from client to server that the *Interest* message has travelled and it is depicted in Equation 2. While the Stretch value tends to 0, requests are solved closed to users and CCN shows its efficiency. $|R|$ refers to the total number of requests.

$$Stretch = \frac{\sum_{i=1}^{|R|} hops\_walked_i}{\sum_{i=1}^{|R|} total\_hops_i} \quad (2)$$

Diversity is a useful metric to calculate the number of distinct elements stored at the caches. It may be seen as a counter of replicas and it assumes that the cache size of nodes are homogeneous. Diversity expresses the ratio of unique

content stored across all the caches (Equation 3). Diversity varies between $[\frac{1}{|N|}, 1]$: if Diversity tends to $\frac{1}{|N|}$, caches contain the same elements and there are many replicas of the same content in the entire network; otherwise, if Diversity tends to 1, the caches store distinct pieces in every cache.

$$Diversity = \frac{len(\bigcup_{n=1}^{|N|} C_n)}{\sum_{n=1}^{|N|} len(C_n)} \quad (3)$$

V. Results

In this section, we compare the caching strategies in the common evaluation scenario according to the presented metrics: Cache Hit, Stretch, Diversity and Complexity.

A. Complexity

By implementing the caching strategies into our simulator, we also compute their complexity and analyze and compare their worst case performance. The complexity for the functions *insert*, *delete* and *lookup* in LRU implementations is $\mathcal{O}(1)$ using a hash table [21]. We analyze the complexity of caching strategies in a standard CCN interaction: an *Interest* message is sent through the network and a *Data* message is received with the answer of the message. We refer to m as the walked hops to retrieve the content and n as the cache size. The walked hop refers to the number of hops made to retrieve the content. The results are presented in the Table III.

As stated before, LCE stores a copy of the *Data* message in every node it has passed. In the best case, the content is found in a CCN cache without forwarding the *Interest* and its complexity is the cost of a lookup operation: $\mathcal{O}(1)$. In the worst case, the *Interest* travels through the whole path (length m) to the origin server. Its complexity is the cost of a lookup operation in every node and then the cost of a store operation in every node: $\mathcal{O}(LCE) = \mathcal{O}(m \times 1 + m \times 1) \approx \mathcal{O}(m)$. The LCE complexity is presented in Table III.

LCD is a special case of LCE. Content is copied in the direct neighbor towards the requester which implies only one insert operation. In contrast with LCE, the cost is reduced to check all the caches in the path and storing only once in the direct neighbor where the content was found (see the second row in Table III).

MAGIC strategy requires minimizing the cost of replacing an element. MAGIC requires to lookup on the caches of the path (m lookup operations) and then to traverse every element of every cache looking for the minimum replacement penalty (m operations of cost n) [9]. The Third row in Table III shows its computational complexity. As caches are expected to be bigger than 10^6 elements, complexity of MAGIC is expected to skyrocket and to decay its performance.

Cache "Less" For More uses information about the topology. This information is calculated off-line, and it has a cost of $\mathcal{O}(|L|^2)$. However, this is calculated only once and then the cost of execution of the algorithm is the same than LCE adding the cost of looking for the best place to cache the content (a comparison made at every node in the path). The complexity is depicted in the fourth row in Table III.

In ProbCache, every node has knowledge to the distance from origin to target server. The content is stored at the best location following a similar approach to the one in Cache "Less" For More. The complexity is similar to the previous one and can be found in the fifth row in Table III.

Caching Strategy	Complexity
Leave Copy Everywhere (LCE)	$\mathcal{O}(m)$
Leave Copy Down (LCD)	$\mathcal{O}(m)$
MAGIC	$\mathcal{O}(m \times n)$
Cache Less For More	$\mathcal{O}(m)$
ProbCache	$\mathcal{O}(m)$

TABLE III: Computational Complexity for Caching Strategies

From Table III, we observe that four caching strategies show a linear cost depending on the length of path to the content and MAGIC has the biggest computational cost. MAGIC is the only caching strategy, which depends on two variables (cache size and path). For example with caches of one million elements, MAGIC requires to run through *m* millions elements while other strategies requires approximately to perform only $3 \times m$ operations.

Figure 2 shows the comparison of the caching strategies in the common evaluation scenario. This figure is divided into 3 lines and 4 columns: Each line stands for three metrics to analyze the caching strategies (Cache Hit, Stretch and Diversity); each column stands for a different content popularity (Zipf α parameter). On each plot, x-axis is the cache size and y-axis is the metric. For each value, each experiment has been performed three times (Table II) and we show the confidence intervals. Every point represents the average of the metric for the three runs and it is averaged with the results of the other topologies.

B. Cache Hit

In the first line of the Figure 2, we compare the Cache Hit of the caching strategies in different environments ranging from low to high popularity scenarios (α=0.65, α=1.1, α=1.5 and α=2.0). We should pay special attention to smaller cache size ratios. The bigger the cache size ratio, the easier to achieve good caching results. However, the goal is to minimize spent resources (cache size) and to maximize caching efficiency. For example, if we consider a catalog of 10^{13} pieces, a cache size ratio of 10^{-6} pieces refers to a cache size of ten millions elements. While a cache size ratio of 10^{-3} refers to ten billion elements.

In terms of Cache Hit, MAGIC and LCD show the best results and outperform the other strategies. In low popularity scenarios, MAGIC performs better with smaller cache sizes and it is overcame by LCD when the cache size increases. For small-size cache, MAGIC exhibits the best performances; however when the cache size increases LCD outperforms other strategies.

LCE and ProbCache show almost the same level of performance. However, ProbCache performs fewer operations than LCE and is therefore a better candidate than LCE (computational complexity).

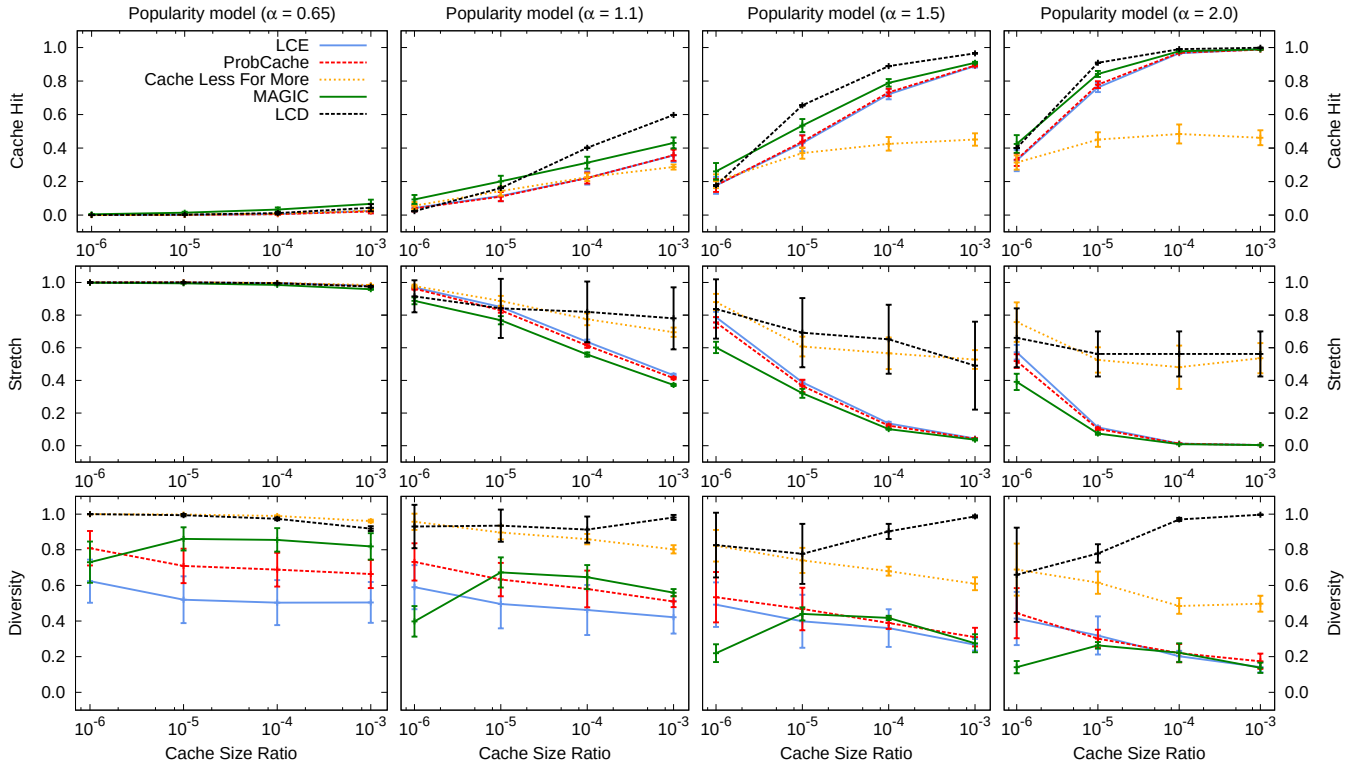


Fig. 2: Comparison of the Caching Strategies

Cache "Less For More" strategy has shown different results. In the scenarios with low popularity ($\alpha = 0.65$ and $\alpha = 1.1$), especially for small-size cache (10^{-6} and 10^{-5}) Cache "Less" For More reaches the same level of performances as other strategies. However, with high popularity and larger cache size, it performs badly compared with others ($\alpha = 1.5$ and $\alpha = 2.0$).

MAGIC and LCD have shown the best Cache Hit performances. Due to its high complexity cost, we believe MAGIC may be used as a boundary function for other caching strategies (see Section V-A). According to our results, LCD should be implemented as replacement for LCE as CCN default caching strategy. LCE and ProbCache present similar results, however ProbCache performs less operations. Finally, we consider Cache "Less For More" strategy is useful in scenarios with low and normal popularity and with cache sizes smaller than 10^{-5}.

C. Stretch

The Second line in the Figure 2 shows the Stretch metric. In low popularity scenarios ($\alpha = 0.65$), all the caching strategies perform badly. Still, MAGIC performs better than the rest of the strategies but its performance is still at low level. With a cache size ratio of 10^{-3}, Stretch is 97% which means the travelled path is only reduced by 3%.

In normal popularity scenarios ($\alpha = 1.1$), for cache size ratio smaller than 10^{-5}, all the strategies show similar performances. For cache size ratio larger than 10^{-5}, we can distinguish two classes of strategies: MAGIC, LCE and ProbCache have the same level of performances and outperform LCD and Cache "Less for More". Overall, regarding the Stretch, MAGIC is the best caching strategy and reduces the length of the path to 88% (cache size 10^{-6}) and to 33% (cache size ratio of 10^{-3}).

In high popularity scenarios ($\alpha = 1.5$ and $\alpha = 2.0$), the level of performance of MAGIC, ProbCache and LCE increases and tends to converge to exactly the same values. LCD and Cache "Less" For More still perform badly and are overcame by 40 percentage points with large cache size ratio (10^{-3}).

MAGIC shows the best results for Stretch. In normal and high popularity scenarios, MAGIC outperforms all the strategies when the cache size is less than 10^{-5}. When the cache size increases, LCE and ProbCache reach the same level of performance as MAGIC.

Note that the complexity of MAGIC is proportional with cache size and its computational cost will be too expensive for large cache size ratios.

D. Diversity

The third line in the Figure 2 shows the Diversity. The caching strategies can be grouped into two classes: High Diversity or Low Diversity. By its design, a caching strategy may generate multiple replicas of the same content, resulting in a Low Diversity into caches. Or, it may limit the number of replicas, resulting in the High Diversity into caches. Thus, MAGIC, ProbCache and LCE belong to the Low Diversity

class, whereas Cache "Less For More" and LCD are in the High Diversity class.

In the Low Diversity class, LCE shows lowest Diversity and can be considered as an appropriate caching strategy if low diversity is expected in the network. Once the popularity or the cache size increase, the gap between the three strategies of this class decreases and their Diversity tends to the same value.

In the High Diversity class, with the low popularity scenario ($\alpha = 0.65$), Cache "Less" For More outperforms LCD. Once the popularity increases ($\alpha \geq 0.65$), LCD becomes the strategy with the highest Diversity and it tends to the maximum value (1.0). In contrast, when popularity increases, Cache "Less" For More Diversity tends to decrease and does not follow the same behavior as LCD.

VI. Summary

A quick look into the results may suggest that MAGIC is the best caching strategy according to Cache Hit and Stretch. However, the analysis of its complexity showed that MAGIC is an expensive caching strategy and its performances into real implementation would be limited. MAGIC can still be used as a boundary function for the further evaluation of future proposals.

Then, choosing the best strategy depends on the level of Diversity expected in the CCN network. In fact, the CCN network may contain diverse content or not : it depends on the needs of the network. For example in disaster scenarios [22], Low Diversity causes high number of replicas and it is an essential property to continue operating properly in the case of failures. Moreover, in some case, multiple replicas (i.e.: Low Diversity) may serve content that is originally located in an unavailable server, while in another case it can be considered as waste of cache space. Otherwise in case of low popularity of content ($\alpha \leq 0.65$), a High Diversity is more appropriate because users are more likely to request distinct content rather than the same content.

For a Low Diversity scenario, ProbCache and LCE appear as good strategies. They also show similar results in terms of Complexity, Cache Hit and Stretch.

For a High Diversity scenario, there is two cases: for small cache size and low popularity (up to 10^{-5} and 1.1 respectively), Cache "Less For More" is the best candidate as it achieves good performances in term of Cache Hit and Stretch while still has a low complexity cost. Otherwise, we suggest resorting to LCD strategy, which overcomes all the other caching strategies.

VII. Conclusion

In this paper, we compare the most relevant CCN caching strategies. We first provide a common evaluation scenario to evaluate the strategies under the same simulation environment. Then, we implemented LCE, LCD, ProbCache, MAGIC and Cache "Less" for More and computed their complexity.

We summarize the results and show in case-by-case basis what caching strategy is appropriate for each scenario. For instance MAGIC outperforms other caching strategy but its computational cost is very high and it can be expensive to be implemented in real CCN nodes. However, MAGIC could be considered as a boundary function for further comparison and evaluation. Even though LCD and Cache "Less for More" are good candidates to be used as caching strategy for CCN, LCE and ProbCache are more appropriate with low Diversity scenarios while LCD and Cache "Less for More" are better candidates with high Diversity. Selecting the best strategy depends on the objectives of the CCN network.

As the accurate evaluation of CCN under a common evaluation scenario is an important topic at the IRTF, we aim to complement this work and contribute to the future deployment of the CCN architecture.

References

[1] 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.

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

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

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

[5] "CCNx project," http://www.ccnx.org

[6] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013.

[7] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," ACM SIGCOMM Workshop ICN 2012.

[8] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," *Comput. Commun.*, vol. 36, no. 7, pp. 758–770, Apr. 2013.

[9] J. Ren, W. Qi, C. Westphal, J. W. Kejie Lu, S. Liu, and S. Wang, "MAGIC: a distributed max-gain in-network caching strategy in information-centric networks," in *IEEE INFOCOM Workshop NOM 2014*.

[10] 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 *ACM SIGCOMM 2013*.

[11] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.

[12] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *IEEE INFOCOM WKSHPS Nomen 2012*.

[13] G. Rossini and D. Rossi, "A dive into the caching performance of content centric networking." in *CAMAD*. IEEE, 2012, pp. 105–109.

[14] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-as cooperative caching for content-centric networks," in *ACM SIGCOMM ICN Workshop 2013*.

[15] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *ACM SIGCOMM ICN 2013 Workshop*.

[16] L. A. Adamic and B. A. Huberman, "Zipf's law and the Internet," *Glottometrics*, vol. 3, 2002.

[17] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *INFOCOM Workshops*, 2012, pp. 310–315.

[18] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *IEEE INFOCOM Workshops 2012*.

[19] E. J. Rosensweig, D. S. Menasch, and J. Kurose, "On the steady-state of cache networks." in *IEEE INFOCOM 2013*.

[20] "https://github.com/mesarpe/socialccnsim

[21] "https://github.com/jlhutch/pylru

[22] I. Psaras, L. Saino, M. Arumaithurai, K. Ramakrishnan, and G. Pavlou, "Name-based replication priorities in disaster cases," in *IEEE Infocom Workshop NOM 2014*.