Caching Policies for In-Network Caching

Zhe Li, Gwendal Simon, Annie Gravey
Institut Mines Telecom - Telecom Bretagne, UMR CNRS 6074 IRISA
Université Européenne de Bretagne, France
{firstname.lastname}@telecom-bretagne.eu

***Abstract*—Recent works on Information Centric Networking enable the exploitation of the caching resources in the new generation of routers (Content Routers or CR). So far, only a basic Least Recently Used (LRU) strategy implemented on every CR has been proposed. More generally, the research community lacks methods for analyzing and evaluating caching policies (other than LRU) in generic multi-cache topologies. In this paper, we provide a model that approximates the hit-ratios of any multi-cache topology for the Least Recently/Frequently Used (LRFU) caching policies, which consist of a spectrum of policies based on a trade-off between recency and frequency. We also present a way to approximate the performances of the network of caches when the input traffic changes. The approximation results can be used to decide suitable policy for CR at different positions in the network topology. With appropriate policy for each single CR, we are able to improve the performances of the whole in-network caching system.**

I. Introduction and Background

The deployment of Internet routers having caching capabilities (CR for Content or Caching Router [1]) is an opportunity to revisit the techniques that are currently used to deliver content in the Internet. A network of CR is in particular a key component of projects related to *information-centric networks*, where requests are no longer routed toward a unique destination and any equipment can act as server [2]. Besides, Internet Service Providers (ISPs) deploy CRs in their networks in order to directly serve their customers, resulting in a shift in the balance of inter-domain traffic and the role of Content Delivery Networks (CDNs). We refer to the management of caches of a network of CRs as *in-network caching*. The research in this area has recently flourished [3]–[7].

The problem addressed in this paper is as follows. Every CR implements a *caching policy*, which uses local information to decide whether a transferred data should be stored, and which stored data should be discarded. Our goal is to determine the caching policy that maximizes the *hit-ratio* of the in-network caching.

Previous works related to information-centric networking have considered only the *Least Recently Used* (LRU) policy. The main advantage of LRU is its simplicity. The implementation of this policy does not affect the routing task of the CR. Other lightweight caching policies have not been considered, principally because the research community misses tools to analyze and evaluate policies for in-network caching.

This work is supported by Vipeer project and Agence Nationale de la Recherche, France

Our contributions are twofold. First, we present an analytical tool that approximates in generic in-network caching systems the performances of the caching policies that are based on the analysis of both recency and frequency of requests. To validate this analytical tool, we compare the theoretical performances of CRs to the one estimated from simulations.

Second, we present a *multi-policy* in-network caching, where every CR implements its own caching policy according to its location in the network. The results obtained with our analytical tool yield a simple method to determine the optimum caching policies for the CRs.

We demonstrate the interest of our multi-policy in-network caching approach by implementing a network of CCNx nodes in the context of a VoD application. As presented in [5], we assume that the operator has reserved a portion of the CRs' caching resources for this application. Compared to the single LRU policy, we show that the multi-policy approach increases the performances (in terms of hit-ratio) of each CR by 15%.

The rest of this paper is organized as follows. Related work about multi-cache analysis is given in §II. Then, we present our simulation based study of LRFU caching performance in §III. The approximation model for multi-cache LRFU caching policies is derived and validated in §IV. Thereafter, in §V we highlight the usefulness of our approximation. §VI extends our basic LRFU approximation so that it can approximate the caching performance with changing object popularity. Finally, the paper is concluded in §VII.

II. Related Works

The theoretical analysis of a generic topology of caches is a challenging topic. The behavior of LRU has been studied in simple topologies like trees [8] but this model is inapplicable for the irregular inter-connected structures of ISP networks. In [9], the authors designed an approximation for a random replacement policy using an approach similar to the one described in [8]. But again, the approach is limited to 2-level cache hierarchies, and cannot be easily extended to mesh or larger tree networks. The authors of [3] developed a mathematical model based on continuous time Markov chains for a single CR. They showed that the performance of a complete multi-cache system can be approximated by their single cache model. However, their model cannot be used to study the performance of each individual cache. Rosenweig *et al.* [10] presented a model that approximates miss ratios for any multi-cache topology when the caching policy is LRU.

This model allows to estimate the performance of each single cache in a multi-cache system. It is based on the single cache model given in [11]. All these analytical models are however specifically designed for the LRU policy.

Our objective is to extend the multi-cache model proposed in [10] by incorporating other caching policies. Among the lightweight caching policies that could be reasonably implemented in a CR, we focus on the well-known *Least Frequently Used* (LFU) policy and the spectrum of policies based on a trade-off between recency and frequency. We use the model of *Least Recently/Frequently Used* (LRFU) caching policy as it has been presented in [12]. We derive an analytical steady-state model for LRFU in order to expand the multi-cache approximation to any caching policy based on both recency and frequency. We also design a technique that allows to approximate the caching performances when the popularity of the requested objects changes over time.

III. Simulation Based Study of LRFU

In this Section, we recall the principles of the LRFU caching policy in §III-A. Then, we detail its performance on both single cache and multi-cache cases.

A. Introduction to LRFU Policy

A cache policy is intended to keep copies of the objects that are more likely to be requested in the future. While the LRU caches the objects that have been more recently requested, the LFU stores the objects that have been more frequently requested. Both LRU and LFU policies suffer from the sidedness of their analysis of the more recent accesses. The LRFU policy [12] aims combining LRU and LFU. Each object is associated with a *Combined Recency and Frequency* (CRF) value, which aims at characterizing the likelihood that it will be accessed in the near future. The computation of CRF uses a function $f(x) = e^{-\gamma x}$ to weigh the importance of the most recent occurrences for an object. The *weighing parameter* γ, which takes its value from 0 to 1, allows a trade-off between recency and frequency such that the more recent accesses become prevalent when γ increases. The computation of the CRF of an object b at time t_b is given below:

$$C_{t_b} = \begin{cases} f(0) + f(t_b - t'_b) * C_{t'_b}, \text{if } b \text{ is known} \\ f(0), \text{if } b \text{ is unknown} \end{cases} \quad (1)$$

where t'_b is the timestamp of the latest access to b [12]. Eq.1 gives more weight to the more recent requests since $f(x)$ is monotonically nonincreasing. On the other hand, $C_{t'_b}$ takes into account the previous accesses so that successive requests contribute to the CRF value. The extreme values 0 and 1 lead to the classical LFU and LRU caching policies respectively.

B. Single Cache Performance

We first simulated the behavior of *one* cache implementing the LRFU caching policies. There were $45{,}000$ available objects in the system. Typically objects are one-minute video segments in a VoD service with 500 movies. The popularity of objects followed a *Zipf distribution* with a skew factor equal to one. Such distribution is an accurate model of movie requests in a typical VoD service [13]. The size of the cache varied from 100 to $1{,}000$, which seems reasonable values, with respect to the limited storage of CRs and the fact that only a part of the storage is reserved for the application.

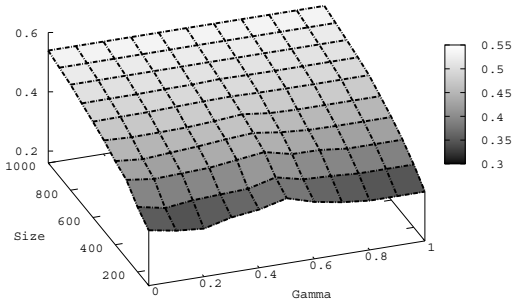

Fig. 1. Single-cache simulation with static object popularity (On the y-axis, absolute hit-ratio produced by simulation.)

In Fig. 1 we display the absolute hit-ratios produced by the simulation. The cache with the size of 100 objects is able to satisfy up to 30% of requests asking for $45{,}000$ objects, depending on the value for γ. When the cache size augments to $1{,}000$ objects, the cache hit-ratio reaches 54%.

Actually, Fig. 1 reveals that for a single cache directly receiving requests from clients, the best configuration is to set γ around 0.5. When the cache capacity is quite limited, *i.e.*, 100 objects, the LRFU with $\gamma = 0.5$ outperforms the worst case $\gamma = 0.2$ about 17.6%. Under the same condition, the configuration $\gamma = 0.5$ works 9.5% and 16.2% better than basic LFU and LRU respectively. On the other hand, we find that for larger caches, the impact of the variation of γ turns to be small. All different configurations of γ yield similar average cache hit-ratio, although $\gamma = 0.5$ still performs slightly better than other configurations. The benefit is about 0.1% comparing with the worst γ configuration.

C. Multi-Cache Performance

We emulated the multi-cache network according to a real backbone topology (European Backbone *Ebone* [14]). We chose 40 nodes interconnected with 108 bidirectional links from the original ISP map. Three nodes are servers storing all objects. In order to emulate the traffic generated by end-users, every CR generated κ requests by round, κ being randomly chosen between 2 and 10. Missed requests were redirected along the shortest path from the origin node to the nearest server. The simulation is $10{,}000$ rounds. Every CR had a cache capacity of 100 objects, which is the worst case according to the results on single cache.

The absolute hit-ratio of the multi-cache simulation is exhibited in Fig. 2. For all configurations of γ, we observe the same prominent disparity between the best and the worst hit-ratio, which ranges from 0.1 to 0.35. But the caching policy with γ at 0.4 and 0.6 performs better than their counterparts

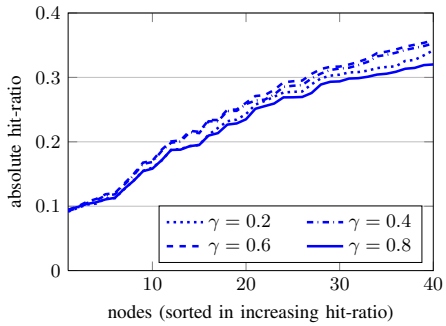

Fig. 2. Multi-cache simulation with static object popularity.

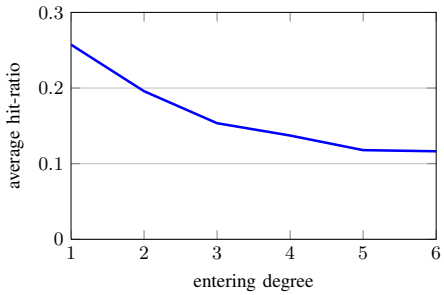

Fig. 3. Impact of the number of incoming request streams.

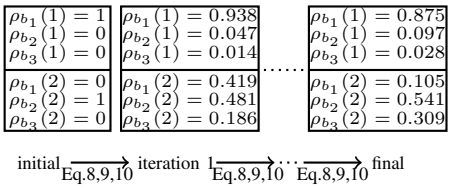

Fig. 4. Example of cooperative cache

with more extreme values of γ for the routers that have a better cache hits. On the contrary, these latter perform slightly better for routers with low cache hits. In §V, we will take advantage of this characteristic to improve the overall in-network caching hit-ratio.

In order to better understand why CRs experience widely different hit ratios, we conjecture that this is due to their "position" on the path between the client and the server. We first define the notion of *entering degree*: given the fact that when a CR cannot meet a request, this request is forwarded to the next CR r on the shortest path to its server, the entering degree of r is the number of shortest paths connecting front-end CRs with servers via r. We consider that all requests from clients to one edge CR as one request flow and we classify CRs by their *entering degree*. Front-end CRs have an entering degree of 1 because they only receive traffic from end-users. A CR with a entering degree larger than 1 treats requests that have been missed by at least one previous CR. In our network configuration, the entering degrees range from 1 (for the Front-end CRs) to 6 (for the well-connected back-end CRs near the servers).

Then, in order to validate our conjecture, we analyze the impact of the entering degree on the hit ratio performances of the CRs in Fig. 3 (only LRU ($\gamma = 1$) policy is shown here although similar results have been obtained for other policies). This figure does indicate that the larger is the entering degree, the lower is the hit-ratio. CRs with a large entering degree that receive requests for less popular objects in the long-tail of the popularity distribution demonstrate the worst average hit-ratio. This indeed validates our conjecture.

IV. Generalized Multi-cache Approximation

We now derive an analytical approximation of a single LRFU cache in §IV-A. More precisely, we approximate the stationary buffer hit probability for each object in the system, *i.e.*, the probability that each object is in the cache. The approximation is based on two assumptions: the input stream conforms to the *Independent Reference Model*, and the incoming requests for objects follow a *Poisson process*. Both assumptions are common in previous works related to cache approximation. We show in §IV-B that these assumptions do not affect the quality of the approximation.

A. Derivation of the analytical approximation

An object is a generic piece of information. For simplicity, we assume that all objects have the same size (for example, an object can represent a video chunk in a VoD application). Let B be the total number of distinct objects and let S be the number of objects a cache can store. We consider an *Independent Reference Model* (IRM), so the successive requests form a sequence of *i.i.d* random variables, each distributed according to the access probability. We note α_b the access request probability of object b. See Table I for the notations.

γ	weighing parameter for LRFU ($\gamma = 0$ for LFU, $\gamma = 1$ for LRU)
C_{t_b}	CRF value of an object b a time t
$\overline{C(b)}$	steady value of the CRF of an object b
B, S	number of objects, capacity of the single cache
α_b, λ_b	access request probability and access rate of b
$P_{bb'}$	probability that b has larger CRF value than b'
P_{br}	probability that the last request was for object b
$\rho_b(s)$	probability that b is at the s-*th* position of the cache
$P_b(s)$	probability that b is within the s first positions of the cache

TABLE I
Notations

We assume that the incoming requests to all objects form a Poisson process. The access rate for object b is defined as $\lambda_b = \lambda \cdot \alpha_b$, where λ is the total request rate. Therefore, the time interval between two successive requests for a given object b follows an exponential distribution with expectation $\frac{1}{\lambda_b}$. Using the classical notation, the distribution for these time intervals relative to object b is given by:

$$g(x; \lambda_b) = \lambda_b e^{-\lambda_b x} \tag{2}$$

Keep in mind that our objective is to derive the probability that a requested object is in the cache. Our approximate derivation relies on three successive steps: i) We approximate the steady CRF value of each object; ii) From the steady CRF value, we derive the probability that an object b is before another object b' in the cache when a request arrives; iii) Then we use *fixed point method* to approximate the probability that each object is in the cache.

The first step of our derivation consists in obtaining an approximate value for the CRF value of an object b. Because

we are considering the steady state value of CRF, we claim that we can approximate it by its expectation:

$$\begin{aligned}\overline{C(b)} \approx E(C_{t_b}) &= E\left(f(0) + C_{t'_b} \cdot e^{-\gamma(t_b - t'_b)}\right) \\ &= 1 + E\left(C_{t'_b} \cdot e^{-\gamma(t_b - t'_b)}\right)\end{aligned}$$

Since the time interval of two successive requests $(t_b - t'_b)$ for b is independent of $C_{t'_b}$, we have:

$$E(C_{t_b}) = 1 + E\left(C_{t'_b}\right) \cdot E\left(e^{-\gamma(t_b - t'_b)}\right) \tag{3}$$

As we assume that C_{t_b} and $C_{t'_b}$ are the steady CRF values of b, we conclude that $E(C_{t_b}) = E(C_{t'_b})$. From Eq.3 we obtain:

$$E(C_{t_b}) = \frac{1}{1 - E(e^{-\gamma(t_b - t'_b)})} \tag{4}$$

On the other side, the expectation of $e^{-\gamma(t_b - t'_b)}$ deduced from Eq.2 is:

$$E(e^{-\gamma(t_b - t'_b)}) = \frac{\lambda_b}{\lambda_b + \gamma} \tag{5}$$

Substituting $E(e^{-\gamma(t_b - t'_b)})$ in Eq.4 by Eq.5, we naturally obtain the approximation of $\overline{C(b)}$:

$$\overline{C(b)} \approx 1 + \frac{\lambda_b}{\gamma} \tag{6}$$

During the second step of our derivation, we compute the probability that, due to a demand for object b, object b' (which is currently in the cache) is pushed down. This probability is approximated by the probability that the new $C_{(t+x)_b}$ is larger than $C_{(t+x)_{b'}}$, where $C_{(t+x)_b} = 1 + e^{-\gamma x} * (1 + \frac{\lambda_b}{\gamma})$ and $C_{(t+x)_{b'}} = 1 + \frac{\lambda_{b'}}{\gamma}$:

$$\begin{aligned}1 + e^{-\gamma x} * (1 + \frac{\lambda_b}{\gamma}) &> 1 + \frac{\lambda_{b'}}{\gamma} \\ x &< \frac{\ln(\frac{\gamma + \lambda_b}{\lambda_{b'}})}{\gamma}\end{aligned} \tag{7}$$

Then the probability for b to possess the larger CRF value than b' is:

$$P_{bb'} = P(x < T) = 1 - e^{-\lambda_b T} \tag{8}$$

where

$$T = \left(\frac{\ln(\frac{\gamma + \lambda_b}{\lambda_{b'}})}{\gamma}\right)^+,$$

and

$$(v)^+ = \begin{cases} v & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases}$$

The third step of our derivation consists in deriving the steady state distribution of the cache's content. Using the previous result, we derive the probability that b is inserted within the first s positions of the cache upon the arrival of a new request by iteratively calculating the following conditional probability:

$$\begin{aligned}P_b^n(s) =& \alpha_b \cdot \left(P_b^{(n-1)}(s) + \sum_{b' \in B, b' \neq b} (\rho_{b'}^{(n-1)}(s) P_{bb'})\right) \\ &+ \sum_{b' \in B, b' \neq b} \left(\alpha_{b'} \cdot P_b^{(n-1)}(s)(1 - P_{b'b})\right)\end{aligned} \tag{9}$$

and the equation:

$$\rho_b^n(s) = P_b^n(s) - P_b^n(s-1), s = 2, 3, \ldots, S, \tag{10}$$

where n is the number of iteration. We do not know $P_b^n(s)$, but starting our iterative computation from an arbitrary cache distribution, the above equations allow to converge towards the steady state distribution of the cache's contents.

In other words, we obtain the probability that each object is in the cache heap by iteratively solving the equations Eq.8, Eq.9 and Eq.10 for each $b \in B$, assuming an arbitrary cache distribution at the first iteration:

$$\begin{aligned}P_{bb'} =& 1 - e^{-\lambda_b T} \\ P_b^n(s) =& \alpha_b \cdot \left(P_b^{(n-1)}(s) + \sum_{b' \in B, b' \neq b} (\rho_{b'}^{(n-1)}(s) P_{bb'})\right) \\ &+ \sum_{b' \in B, b' \neq b} \left(\alpha_{b'} \cdot P_b^{(n-1)}(s)(1 - P_{b'b})\right) \\ \rho_b^n(s) =& P_b^n(s) - P_b^n(s-1), s = 2, 3, \ldots, S,\end{aligned}$$

The iteration is stopped when the probability distribution meets a predefined precision threshold. In our implementation, we take the mean square error of the probabilities calculated by two consecutive iterations as the halting criterion. When the mean square error is less than the predefined value ϵ, we stop the iteration. This procedure has converged quickly in all the tests we made, although this is not a formal proof for convergence.

The iterative computation process is illustrated by an small example with three objects and a 2-objects cache in Fig.4. The identifier of an object is given according to its access rate (*i.e.*, b_1 is the object with the largest access rate). The initial state is given below:

$$\rho_b^{(0)}(s) = \begin{cases} 1 & \text{if } b \text{ has the } s\text{-th largest access rate} \\ 0 & \text{otherwise.} \end{cases}$$

Please note that the CRF value of objects cannot be computed when $\gamma = 0$, that is, when the LRFU policy becomes LFU. This drawback is not surprising as the CRF value degenerates to the total number of accesses when $\gamma = 0$ and $f(x) = 1$. We propose to set the CRF of an object b as $\overline{C(b)} = P_{br}$ when the caching policy is LFU. Indeed, P_{br} corresponds to the ratio of requests for b among all requests.

This approximation also presents limitations to treat some extreme case. For example, when access rates for all the objects in the cache system are identical, this approximation is unable to decide which object has the higher probability to reside at the first position of the cache. This extreme case

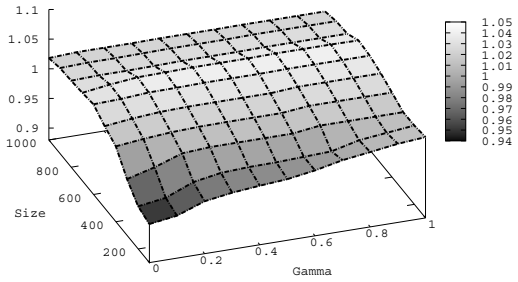

Fig. 5. Comparison of approximation and simulation for a single cache with static object popularity (On the y-axis, ratio of hit-ratio approximation to hit-ratio simulation. The same in Fig. 6 and Fig. 11)

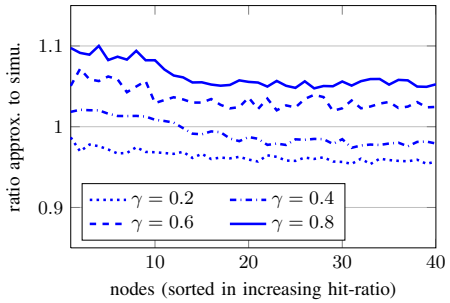

Fig. 6. Multi-cache approximation performance with static object popularity.

may theoretically occur, however, the tie is unlikely to exist in reality. Hence, the shortcoming does not impact the reliability of the approximation when it is used to approximate the performance of a practical system.

This approximation model for a single LRFU cache policy, which computes the probability of existence of each object in the cache, is compatible with the multi-cache model proposed in [10]. Therefore, the algorithms proposed in this latter model can be directly used to obtain an approximation of any LRFU policy in any multi-cache topology.

B. Validation of the approximate model

We show here that the hit-ratio obtained from our approximate model matches the hit-ratio obtained by simulation.

Fig. 5 shows the ratio of the hit-ratio of the approximation to the hit-ratio of the simulation in the case of a single cache. The closer to 1, the better is the approximation. Fig. 5 shows than the approximation model closely matches the simulation results. The error is larger than 5% in only one situation: when LRFU is close to LFU and the cache size is small. Otherwise, the error is less than 4%. Except for the smallest cache size, where ratio varied from -5.5% for $\gamma = 0$ to $+1.6\%$ for $\gamma = 1$, the impact of γ on the approximation ratio was negligible.

We think that the approximation model overestimates the single cache performance because the probability of each object is larger than 0 in the approximation, whereas most of the unpopular objects experience a null hit-ratio in the simulation.

We now validate the multi-cache LRFU approximation. The network topology and the tested parameters are the same described in §III-C. Fig. 6 shows the ratio of the hit-ratio of the approximation to the hit-ratio of the simulation for all routers and four different values of γ, ranging from 0.2 to 0.8. Despite we were in the worst scenario for cache size (100), the error due to approximation was never more than 10% for a given CR. The error was below 5% for all CRs when $\gamma = 0.2$ and $\gamma = 0.4$, and for 36 out of 40 CRs when $\gamma = 0.6$. Moreover, for a given γ, the approximation performed uniformly on all routers. The CRs with the lowest hit-ratio were affected by a larger ratio degradation than the CRs with the highest hit-ratio, but this degradation difference stayed reasonable (less than 5%).

V. Optimizing LRFU in Multi-cache Network

To our best knowledge, the global optimization of a multi-cache network is still an open problem. Although the routers which directly receive requests from end-users ("front-end" CRs) work as classic caches, the behavior of "back-end" CRs, which are between the front-end CRs and the server, have to serve an incoming traffic that is expunged from the most popular requests, which makes their behavior hard to assess and control. Usually, all routers apply the same caching policy, irrespective of their location in the network. In the present Section, we assess the potential gain of varying the caching policy according to the location of the CRs. In our context, considering different caching policies for different CRs consists in selecting different values of γ for CRs with various entering degree.

A. Optimizing cache policies on the Ebone example

We assess the efficiency of various policies through our approximate analytical model of cache behavior. Fig. 7 gives the value of the parameter γ for which a CR with a given entering degree experiences the best hit-ratio in the Ebone network topology considered previously. The front-end CRs reach their best hit-ratio for $\gamma = 0.53$, which is consistent with the results originally presented in [12]. When the entering degree grows, the best γ increases and ultimately, the best caching policy is LRU ($\gamma = 1$) for CRs with entering degree of 6.

This shows indeed that CRs with different entering degree reach their best performances with various values of γ. This suggests that, instead of running the same policy on all CRs, one should carefully select the value of γ for CRs at distinct positions in the network, so that the LRFU with suitable γ value can maximize the performance of each single cache and thus improve the performance of the multi-cache network.

B. Relaxing the Independence assumption

Keep in mind that the the analytical approximation relies on assuming that successive requests are independent and occur as a Poisson process. In a realistic network, this is not the

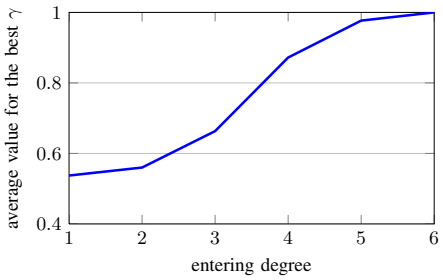

Fig. 7. Best configuration of γ for CRs with different entering degree

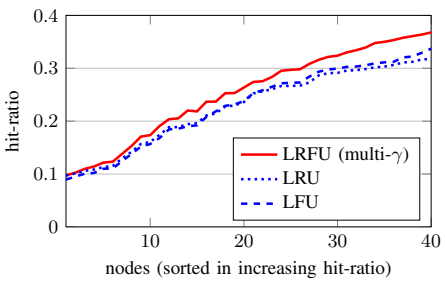

Fig. 8. Hit ratio for different cache policies

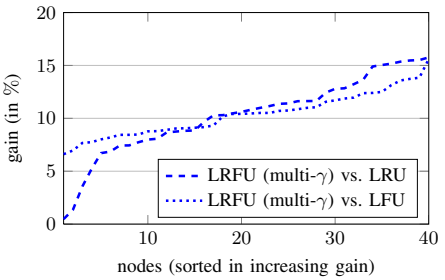

Fig. 9. Gain in terms of hit ratio of LRFU (multi-γ) versus LRU and LFU

case. In the present Section, we shall simulate a network (the Ebone [14]) that implements in-network caching, and is used to support a Video-on-demand (VoD) service.

Servers initially publish all segments for 500 available videos. The size of each video varies uniformly from 60 to 120 segments, so the total number of segments is around 45,000. User behavior (number of users to activate and the daily access pattern) follows measurement results from [13]. Once a client is activated, it chooses a video based on a Zipf law with the skew factor equal to one. The duration for each session is as follows: 50% of sessions end last less than 10 minutes, 75% less than 25 minute, 90% less than 50 minutes, and the remaining sessions last till the end of the video.

The LRFU policy is implemented using the open source CCNx demo. Our modified CCNx prototype is deployed on 40 machines with dual 2.70 GHz Pentium processor and 4 GB RAM. Each machine uses a Ubuntu 10.04 system and is connected to a switch via 100 Mb/s Ethernet card. Every machine works as a router. Among the 40 routers, 20 of them act as edge routers, with the responsibility to emit the requests from 1,000 end users, and 3 routers are servers. The cache capacity of every router is limited to 100 segments.

The γ value is selected according to the results given in Fig. 7 and the topology of the network. For the 20 edge routers, γ is set to 0.53, and the 3 CRs connected with servers used directly the LRU policy. Then, the other 17 intermediate CRs are classified according to their entering degree. Each group of CRs is assigned respectively 4 different γ values: 0.56, 0.66, 0.87, 0.98. This policy is denoted "LRFU multi-γ". We run

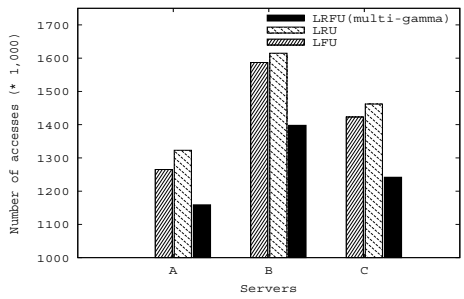

Fig. 10. Number of times each server is accessed

our simulation for 10,000 minutes, *i.e.* about one week.

We consider two aspects of the multi-cache's performance: (i) **The single cache hit-ratio** which measures how often the requested object is within a cache. (ii) **The number of access to servers** which measures the load offered to the servers (keep in mind that using in-network caching aims at reducing this load).

Fig. 8 compares the hit-ratio for three policies (LRU, LFU and LRFU multi-γ). As expected, front-end CRs achieve a higher hit-ratio than back-end CRs for all policies. For edge routers (CRs 21-40) LFU outperforms LRU whereas LRU is slightly better than LFU for the back-end routers. However, LRFU multi-γ is significantly better than both LRU and LFU, especially for front-end CRs. The gain, relative to hit ratio, yielded by LRFU multi-γ is explicitly displayed in Fig. 9. The highest gain in terms of hit ratio reaches 16% comparing with both LRU and LFU policies.

The amount of servers' access is shown in Fig. 10. LFU is slightly more efficient than LRU for the 3 servers, but is significantly less efficient than LRFU multi-γ. In-network caching system using LRFU multi-γ outperforms LRU by at least 15% in terms of servers' access.

This simulation study confirms the results obtained in SectionV-A, which rely on independant requests. In this more realistic scenario where successive demands are not independent since they correspond to successive chunks of the same video, it is still beneficial to finely tune the γ value according to the location of the CRs relative to users and servers.

VI. Approximation Under Dynamic Popularity

On-demand streaming applications, especially catch-up TV, exhibit frequent changes in the popularity of objects. Our approximate model, as all other studies related to caching approximation consider a static popularity distribution.

In the present Section, we slightly modify our approximate model in order to take into account the dynamic popularity changes. We assume that the popularity for the objects periodically change, but that the period duration is long enough for the system to reach a new equilibrium before a new popularity change.

The modified single cache approximation model is obtained as follows: the steady CRF value $\overline{C(b)}$ of an object b, which

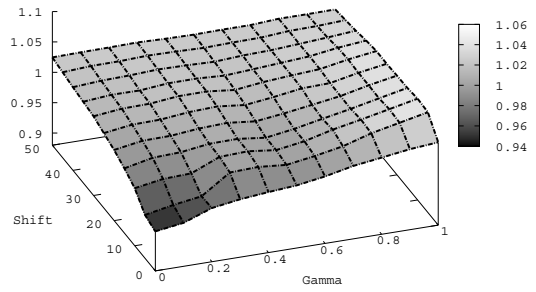

Fig. 11. Comparison of approximation and simulation for a single cache with dynamic object popularity (On the y-axis, ratio of hit-ratio approximation to hit-ratio simulation).

records the history of b's access pattern, is computed as in Section IV-A for a given set of popularity values whereas $P_{bb'}$, probability b pushes down b' is computed using the modified value for the popularity of b.

$$P_{bb'} = P(x < T) = 1 - e^{-\mu_b T}, \tag{11}$$

where μ_b is the new value for the access rate of b, whereas λ_b, the old value for the access rate of b is used to compute T.

We now validate our approximation under dynamic popularity context. The model for modifying the objects' popularity is the following: we first sort all objects in ascending order of their access rates, then we shift all access rates from objects to their immediate predecessor objects in the sorted list. Let consider that object b_i has the ith largest access rate λ_{b_i}. After the shift, μ_{b_i} will be equal to $\lambda_{b_{i+1}}$, which was the previous access rate for b_{i+1}. We iterate this a number of times that we express through a percentage j of the number of objects B. That is, a $j\%$-shift corresponds to a popularity change where object i has taken the access rate of object $i + j * B \mod B$. In other words, $j\%$ of the most unpopular objects have become the most popular ones. This evolution is believed to be particularly suitable for catch-up TV, where the most recent video segments are the most popular, and all other segments have a strictly decreasing popularity [15]. Moreover, since our approximation model requires a fixed number of objects, this algorithm allows to mimick the creation of objects by transforming objects with null popularity in popular objects.

Fig. 11 dhows the ratio of hit-ratio of approximation to hit-ratio of simulation. The error was again below 6% for all configurations except the extreme values $\gamma = 0$ and $\gamma = 1$ when the traffic was static. We observe that our approximation model performs especially well when the traffic changes are more brutal, a 50%-shift being a major disruption in the popularity of objects. In such configuration, the error is less than 3% whatever the value of γ.

VII. Conclusion

Research on Information-Centric Networks currently lacks of tools for the analysis of network performance and delivered QoS. In this paper, we address the evaluation of caching policies for in-network caching, which we believe is a major topic. We provide an analytical tool for the approximation of cache content distribution and hit-ratio for any multi-cache topology and for any LRFU caching policies based on both recency and frequency. This approximation model allows the analysis of a range of caching policies that fit ICN requirements.

Then, we show the advantage of fine-tuning the cache management policy according to the location of CRs relative to users and servers. This advantage is measured in terms of both hit ratio (which improves latency) and servers's load (which reduces network's costs). We thus demonstrate that finding suitable policy for each single cache improves the overall performance of the in-network cache system.

This paper is a positive step in the analysis and the understanding of network of caches, with the promises of better network resource usage.

Future works include an in-depth analysis of the trade-off between caching and bandwidth usage, including the optimization of the caches size, location and management policies. This analysis should be carried out for both static and dynamic objects popularity.

References

[1] U. Lee, I. Rimac, and V. Hilt, "Greening the internet with content-centric networking," in *ACM e-Energy Conference*, 2010.
[2] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *ACM CoNEXT*, 2009.
[3] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and Evaluation of CCN-caching Trees," in *IFIP Networking*, 2011.
[4] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental evaluation of storage management in content-centric networking," in *IEEE ICC*, 2011.
[5] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," Orange Labs/Alcatel-Lucent, Tech. Rep., 2011.
[6] W. Wong, M. Giraldi, M. F. Magalhães, and J. Kangasharju, "Content routers: Fetching data on network path," in *IEEE ICC*, 2011.
[7] Z. Li and G. Simon, "Time-Shited TV in Content Centric Networks: the Case for Cooperative In-Network Caching," in *IEEE ICC*, 2011.
[8] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, 2002.
[9] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for lru cache performance," *arXiv.org*, Feb. 2012.
[10] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *IEEE INFOCOM*, 2010.
[11] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *ACM SIGMETRICS*, 1990.
[12] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies," *IEEE Trans. on Computers*, vol. 50, no. 12, 2001.
[13] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 4, 2006.
[14] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *ACM SIGCOMM*, 2002.
[15] Y. Liu and G. Simon, "Distributed Delivery System for Time-Shifted Streaming System," in *IEEE LCN*, 2010.