A Unified Approach to the Performance Analysis of Caching Systems

MICHELE GARETTO, Università di Torino
EMILIO LEONARDI and VALENTINA MARTINA, Politecnico di Torino

We propose a unified methodology to analyze the performance of caches (both isolated and interconnected), by extending and generalizing a decoupling technique originally known as Che's approximation, which provides very accurate results at low computational cost. We consider several caching policies (including a very attractive one, called k-LRU), taking into account the effects of temporal locality. In the case of interconnected caches, our approach allows us to do better than the Poisson approximation commonly adopted in prior work. Our results, validated against simulations and trace-driven experiments, provide interesting insights into the performance of caching systems.

Categories and Subject Descriptors: C.2.1 [**Network Architecture and Design**]: Distributed Networks; C.4 [**Performance of Systems**]: Modeling Techniques

General Terms: Performance

Additional Key Words and Phrases: Caching, content delivery networks, information-centric networking

ACM Reference Format:
Michele Garetto, Emilio Leonardi, and Valentina Martina. 2016. A unified approach to the performance analysis of caching systems. ACM Trans. Model. Perform. Eval. Comput. Syst. 1, 3, Article 12 (May 2016), 28 pages.
DOI: http://dx.doi.org/10.1145/2896380

1. INTRODUCTION AND PAPER CONTRIBUTIONS

In the past few years, the performance of caching systems, one of the most traditional and widely investigated topics in computer science, has received a renewed interest by the networking research community. This revival can be essentially attributed to the crucial role played by caching in new content distribution systems emerging on the Internet. Thanks to an impressive proliferation of proxy servers, Content Delivery Networks (CDN) represent today the standard solution adopted by content providers to serve large populations of geographically spread users [Jiang et al. 2012]. By caching contents close to the users, we jointly reduce network traffic and improve user-perceived experience.

The fundamental role played by caching systems on the Internet goes beyond existing CDNs, as a consequence of the gradual shift from the traditional host-to-host communication model to the new host-to-content paradigm. A novel Information-centric Networking (ICN) architecture has been proposed for the future Internet to better respond to the today and future (according to predictions) traffic characteristics [Jacobson et al.

This work is supported by the AMALFI project, Università di Torino/Compagnia di San Paolo. A preliminary version of this article was presented at IEEE INFOCOM 2014.
Authors' address: M. Garetto, Dipartimento di informatica, C.so Svizzera 185, 10149, Torino, Italy; email: michele.garetto@unito.it; V. Martina and E. Leonardi, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy; emails: lavale.martina@gmail.com, leonardi@tlc.polito.it.

DOI: http://dx.doi.org/10.1145/2896380

2009]. In this architecture, caching becomes a ubiquitous functionality available at each router.

For these reasons, it is of paramount importance to develop efficient tools for the performance analysis of large-scale interconnected caches for content distribution. Evaluating the performance of cache networks is hard, considering that the computational cost to exactly analyze just a single Least Recently Used (LRU) cache grows exponentially with both the cache size and the number of contents [King 1971; Dan and Towsley 1990]. Nevertheless, several approximations have been proposed over the years [Dan and Towsley 1990; Che et al. 2002; Fricker et al. 2012a; Rosensweig et al. 2010; Gallo et al. 2012; Bianchi et al. 2013] that can accurately predict cache performance at an affordable computational cost.

The main drawback of existing analytical techniques is their rather limited scope. Many techniques target only specific caching policies (mainly LRU and FIFO) under simplifying traffic conditions (most previous work relies on the Independent Reference Model [Coffman and Denning 1973]), while the analysis of cache networks has only recently been attempted (essentially for LRU). See related work in Section 6.

The main contribution of our work is to show that the decoupling principle underlying one of the approximations suggested in the past (the so-called Che's approximation) has much broader applicability than the particular context in which it was originally proposed—that is, a single LRU cache under Independent Reference Model (IRM) traffic—and can actually provide the key to developing a general methodology to analyze a variety of caching systems.

In particular, in this article, we show how to extend and generalize the decoupling principle of Che's approximation along three orthogonal directions: (i) a much larger set of caching algorithms than those analyzed so far (under Che's approximation), implementing different insertion/eviction policies (including a multistage LRU scheme, LRU with probabilistic insertion, FIFO, and RANDOM); (ii) a more general traffic model than the traditional IRM, to capture the effects of temporal locality in the requests arrival process (we consider a general renewal traffic model for all the caching policies mentioned earlier); (iii) a more accurate technique to analyze interconnected caches that goes beyond the standard Poisson assumption adopted so far, and permits also considering smart replication strategies (such as leave-copy-probabilistically and leave-copy-down).

Although, in this article, we cannot analyze all possible combinations of these extensions, we provide sufficient evidence that a unified framework for the performance analysis of caching systems is possible under Che's approximation at low computational cost. Our results for the considered systems turn out to be surprisingly good when compared to simulations (model predictions can hardly be distinguished from simulation results on almost all plots).

Furthermore, under the small-cache regime (i.e., cache size small with respect to the content catalogue size), which is of special interest for ICN, our expressions can be further simplified, leading to simple closed-form formulas for the cache-hit probability, revealing interesting asymptotic properties of the various caching policies. The insights gained from our models are also (qualitatively) confirmed by trace-driven experiments.

To the best of our knowledge, we are the first to propose a unified, simple, and flexible approach that can be used as the basis for a general performance evaluation tool for caching systems.

This article extends the previous conference version in several respects: (i) our modeling approach has been generalized and successfully applied to cache networks with general (mesh) topology, (ii) new material concerning the asymptotic behavior of some of the considered caching policies has been added, and (iii) several parts of have been modified to improve the overall clarity.

2. SYSTEM ASSUMPTIONS

2.1. Traffic Model

We first recall the so-called Independent Reference Model (IRM), which is *de facto* the standard approach adopted in the literature to characterize the pattern of object requests arriving at a cache [Coffman and Denning 1973]. The IRM is based on the following fundamental assumptions: (i) users request items from a fixed catalogue of M object; and (ii) the probability p_m that a request is for object m, $1 \leq m \leq M$, is constant (i.e., the object popularity does not vary over time) and *independent* of all past requests, generating an i.i.d. sequence of requests.

By construction, the IRM completely ignores all temporal correlations in the sequence of requests. In particular, it does not take into account an important feature often observed in real content request traces, typically referred to as *temporal locality*: requests for a given content become denser over short periods of time. The important role played by temporal locality, especially its beneficial effect on cache performance, is well known in the context of computer memory architecture [Coffman and Denning 1973] and Web traffic [Fonseca et al. 2003]. Several extensions of IRM have been proposed to reproduce content temporal locality [Coffman and Denning 1973; Fonseca et al. 2003; Almeida et al. 1996; Jin and Bestavros 2000; Traverso et al. 2013; Garetto et al. 2015]. The majority of the proposed approaches [Coffman and Denning 1973; Fonseca et al. 2003; Almeida et al. 1996; Jin and Bestavros 2000; Garetto et al. 2015] share with the IRM the following two assumptions: (i) the content catalog is fixed; and (ii) the request process for each content is stationary (typically, it is assumed to be either a renewal process or a semi-Markov-modulated Poisson process). Recently, a new traffic model, named the Shot Noise Model (SNM), has been proposed [Traverso et al. 2013] as a viable alternative to traditional traffic models to capture macroscopic effects related to content popularity dynamics. The basic idea of the SNM is to represent the overall request process as the superposition of a potentially infinite population of independent inhomogeneous Poisson processes (shots), each referring to an individual content. The definition of analytical models for the evaluation of cache performance under the SNM [Traverso et al. 2013; Olmos et al. 2014], however, is significantly challenging, as discussed in Garetto et al. [2015], especially when non-LRU caches and networks of caches are analyzed. Moreover, in Garetto et al. [2015], it has been shown that the performance of caching systems under the SNM traffic model can predicted with high accuracy by adopting a fixed-size content catalogue, and modeling the arrival process of each content by a renewal process with a specific interrequest time distribution.

For these reasons, in this article, we will consider the following traffic model, which generalizes the classical IRM. The request process for every content m is described by an independent renewal process with assigned interrequest time distribution. Let $F_R(m,t)$ be the cdf of the interrequest time t for object m. The average request rate λ_m for content m is then given by $\lambda_m = 1/\int_0^\infty (1-F_R(m,t))\,\mathrm{d}t$. Let $\Lambda = \sum_{m=1}^{M} \lambda_m$ be the global arrival rate of requests. Note that, by adopting an object popularity law analogous to the one considered by the IRM, we also have that $\lambda_m = \Lambda p_m$.

In a particular case, our traffic model reduces to the classical IRM when interarrival request times are independently, exponentially distributed, so that requests for object m are generated according to a homogeneous Poisson process of rate λ_m. In the following, we will refer to our generalized traffic model as *renewal* traffic.

2.2. Popularity Law

Traffic models such as the IRM (and its generalizations) are commonly used in combination with a Zipf-like law of object popularity, which is frequently observed in traffic

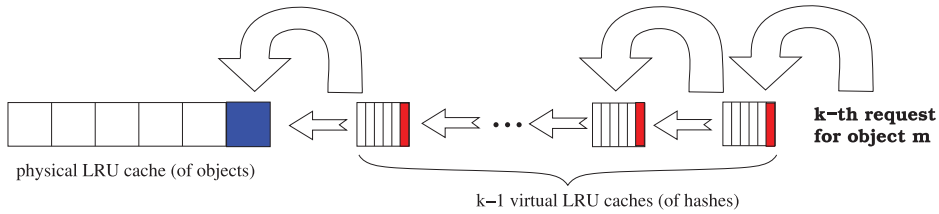

Fig. 1. Illustration of the k-LRU policy.

measurements and widely adopted in performance evaluation studies [Breslau et al. 1999; Cha et al. 2009].

In its simplest form, Zipf's law states that the probability to request the ith most popular item is proportional to $1/i^{\alpha}$, for which the exponent α depends on the considered system (especially on the type of objects), and plays a crucial role in the resulting cache performance [Fricker et al. 2012a]. Estimates of α reported in the literature for various kinds of systems range between .65 and 1 [Fricker et al. 2012b].

In our work, we will consider a simple Zipf's law as the object popularity law, although our results hold in general, that is, for any given distribution of object request probabilities $\{p_m\}_m$.

2.3. Policies for Individual Caches

There exists a tremendous number of different policies to manage a single cache, which differ either for the insertion or for the eviction rule. We will consider the following algorithms as a representative set of existing policies:

—**LFU:** The Least Frequently Used (LFU) policy statically stores in the cache the C most popular contents (assuming that their popularity is known *a priori*); LFU is known to provide optimal performance under IRM.
—**LRU:** Upon arrival of a request, an object not already stored in the cache is inserted into it. If the cache is full, to make room for a new object, the *Least Recently Used (LRU)* item is evicted, that is, the object that has not been requested for the longest time.
—**q-LRU:** It differs from LRU for the insertion policy: upon arrival of a request, an object not already stored in the cache is inserted into it with probability q. The eviction policy is the same as for LRU.
—**FIFO:** It differs from LRU for the eviction policy: to make room for a new object, the item inserted the longest time ago is evicted. Note that this scheme differs from LRU in this respect: requests finding an object in the cache do not "refresh" the arrival time associated to it.
—**RANDOM:** It differs from LRU for the eviction policy: to make room for a new object, a random item stored in the cache is evicted.
—**k-LRU:** This strategy provides a clever insertion policy by exploiting the following idea: before arriving at the (physical) cache that is storing actual objects, indexed by k, requests have to advance through a chain of $k-1$ (virtual) caches put in front of it, acting as filters, which store only object pointers performing caching operations on them (see Figure 1). Specifically, upon arrival of a request, a content/pointer can be stored in cache $i > 1$ only if its pointer is already stored in cache $i-1$ (i.e., the arrival request has produced a hit in cache $i-1$). The eviction policy at all caches is

LRU. Note that this policy[1] can be seen as a generalization of the two-stages policy proposed in Johnson and Shasha [1994], which they called LRU-2Q.

—**k-RANDOM:** It works exactly like k-LRU, with the only difference being that the eviction policy at each cache is RANDOM.

LRU has been widely adopted since it provides good performance while being reasonably simple to implement. RANDOM and FIFO have been considered as viable alternatives to LRU in the context of ICN, as their hardware implementation in high-speed routers is even simpler. The q-LRU policy and multistage caching systems similar to our k-LRU have been proposed in the past to improve the performance of LRU by means of a better insertion policy. We have chosen q-LRU in light of its simplicity, and the fact that it can be given an immediate interpretation in terms of probabilistic replication for cache networks (see next section). The main strength of k-LRU, instead, resides in the fact that it requires just one traffic-independent parameter[2] (the number of caches k), providing significant improvements over LRU even for very small k (much of the possible gain is already achieved by $k = 2$).

2.4. Replication Strategies for Cache Networks

In a system of interconnected caches, requests producing a miss at one cache are typically forwarded along one or more routes to repositories storing all objects. After the request eventually hits the target, we need to specify how the object gets replicated back in the network, in particular, along the route traversed by the request. We will consider the following mechanisms [Rossini and Rossi 2014]:

—**Leave-copy-everywhere (LCE):** The object is sent to all caches of the backward path.

—**Leave-copy-probabilistically (LCP):** the object is sent with probability q to each cache of the backward path.

—**leave-copy-down (LCD):** The object is sent only to the cache preceding the one in which the object is found (unless the object is found in the first-visited cache).

Note that LCP, combined with standard LRU at all caches, is the same as LCE combined with q-LRU at all caches.

3. CHE'S APPROXIMATION

We briefly recall Che's approximation for LRU under the classical IRM [Che et al. 2002]. Consider a cache capable of storing C objects. Let $T_C(m)$ be the time needed before C *distinct* objects (not including m) are requested by users. Therefore, $T_C(m)$ is the *cache eviction time* for content m, that is, the time since the last request after which object m will be evicted from the cache (if the object is not requested again in the meantime).

Che's approximation assumes $T_C(m)$ to be a constant independent of the selected content m. This assumption has been given a theoretical justification recently in Fricker et al. [2012a], in which it is shown that, under a Zipf-like popularity distribution, the coefficient of variation of the random variable representing $T_C(m)$ tends to vanish as the cache size grows. Furthermore, the dependence of the eviction time on m becomes

[1]In the most general case, one could individually specify the size of all caches along the chain. For simplicity, in this article, we will restrict ourselves to the case in which all caches have the same size (expressed either in terms of objects or pointers), since numerical explorations suggest that no significant performance gains can be obtained by tuning the sizes of individual caches.

[2]More sophisticated insertion policies, such as the persistent-access-caching algorithm [Jelenković and Radovanović 2008], obtain a filtering effect similar to k-LRU but require more parameters that are not easy to set, requiring *a priori* knowledge of the traffic characteristics.

negligible when the catalogue size is sufficiently large. For completeness, we note that an indirect proof of Che's approximation asymptotic validity has been provided earlier in Jelenković and Kang [2008] for $\alpha > 1$.

The reason why Che's approximation greatly simplifies the analysis of caching systems is because it allows decoupling of the dynamics of different contents: interaction among the contents is summarized by T_C, which acts as a single primitive quantity representing the response of the cache to an object request.

Thanks to Che's approximation, we can state in more detail that an object m is in the cache at time t if and only if a time smaller than T_C has elapsed since the last request for object m, that is, if at least one request for m has arrived in the interval $(t - T_c, t]$. Under the assumption that requests for object m arrive according to a Poisson process of rate λ_m, the time-average probability $p_{\text{in}}(m)$ that object m is in the cache is then given by

$$p_{\text{in}}(m) = 1 - e^{-\lambda_m T_c}. \tag{1}$$

As an immediate consequence of PASTA property for Poisson arrivals, observe that $p_{\text{in}}(m)$ also represents, by construction, the hit probability $p_{\text{hit}}(m)$, that is, the probability that a request for object m finds object m in the cache.

Considering a cache of size C, by construction:

$$C = \sum_m \mathbb{I}_{\{m \text{ in cache}\}}.$$

After averaging both sides, we obtain the following:

$$C = \sum_m \mathbb{E}\left[\mathbb{I}_{\{m \text{ in cache}\}}\right] = \sum_m p_{\text{in}}(m). \tag{2}$$

The only unknown quantity in this equality is T_C, which can be obtained with arbitrary precision by a fixed-point procedure. The average hit probability of the cache is

$$p_{\text{hit}} = \sum_m p_m \, p_{\text{hit}}(m). \tag{3}$$

4. EXTENSIONS FOR SINGLE CACHE

We will show in the next sections that Che's idea of summarizing the interaction among different contents by a single variable (the cache eviction time) provides a powerful decoupling technique that can also be used to predict cache performance under *renewal* traffic as well as to analyze policies other than LRU.

4.1. LRU Under *Renewal* Traffic

The extension of Che's approximation to the *renewal* traffic model is conceptually simple, although it requires some care. Observe that, under a general request process, we cannot apply PASTA anymore, identifying $p_{\text{in}}(m)$ with $p_{\text{hit}}(m)$. To compute $p_{\text{in}}(m)$, we can still consider that an object m is in the cache at time t if and only if the last request arrived in $[t - T_C, t)$. This requires that the *age* since the last request for object m is smaller than T_C:

$$p_{\text{in}}(m) = \hat{F}_R(m, T_C),$$

where $\hat{F}_R(m, t) = \lambda_m \int_0^t (1 - F_R(m, \tau))\, \mathrm{d}\tau$ is the cdf of the *age* associated to object-m interrequest time distribution.

On the other hand, when computing $p_{\text{hit}}(m)$, we implicitly condition on the fact that a request arrives at time t. Thus, the probability that the previous request occurred

in $[t - T_C, t)$ equals the probability that the last interrequest time does not exceed T_C, yielding:

$$p_{\text{hit}}(m) = F_R(m, T_C).$$

4.2. q-LRU under IRM and *Renewal* Traffic

We now analyze the q-LRU policy (LRU with probabilistic insertion), considering first the simpler case of IRM traffic. In this case, $p_{\text{in}}(m)$ and $p_{\text{hit}}(m)$ are equal by PASTA.

To compute $p_{\text{in}}(m)$, we exploit the following reasoning: an object m is in the cache at time t provided that (i) the last request arrived at $\tau \in [t - T_C, t)$ and (ii) either at τ^- object m was already in the cache or its insertion was triggered by the request arriving at τ (with probability q). We obtain the following:

$$p_{\text{hit}}(m) = p_{\text{in}}(m) = (1 - e^{-\lambda_m T_C})[p_{\text{in}}(m) + q(1 - p_{\text{in}}(m))]. \tag{4}$$

Solving this expression for $p_{\text{in}}(m)$, we get that

$$p_{\text{hit}}(m) = p_{\text{in}}(m) = \frac{q(1 - e^{-\lambda_m T_C})}{e^{-\lambda_m T_C} + q(1 - e^{-\lambda_m T_C})}. \tag{5}$$

Under *renewal* traffic, $p_{\text{in}}(m)$ and $p_{\text{hit}}(m)$ differ by the same token considered for LRU. Repeating the same arguments as before, we get that

$$p_{\text{hit}}(m) = F(m, T_C)[p_{\text{hit}}(m) + q(1 - p_{\text{hit}}(m))], \tag{6}$$

which generalizes Equation (4). The *age* distribution must be used instead to compute $p_{\text{in}}(m)$:

$$p_{\text{in}}(m) = \hat{F}(m, T_C)[p_{\text{hit}}(m) + q(1 - p_{\text{hit}}(m))]. \tag{7}$$

Regarding the q-LRU policy, Che's approximation allows one to establish the following interesting property as $q \to 0$, whose proof is reported in Appendix A (IRM case) and Appendix B (non-IRM case).

THEOREM 4.1. *The q-LRU policy tends asymptotically to LFU as the insertion probability goes to zero both under IRM and under renewal traffic under the following conditions: for any m_1 and m_2 with $\lambda_{m_1} < \lambda_{m_2}$, either $\lim_{t\to\infty} \frac{1-F(m_1,t)}{1-F(m_2,t)} = \infty$ or a T can be found such that $1 - F(m_1, T) > 0$ and $1 - F(m_2, T) = 0$.*

Remark. Note that this condition is satisfied whenever $F(m, t)$ has an exponential tail, that is, $F(m, t) \approx e^{-\alpha_m t}$ with parameter α_m monotonically dependent on the average rate λ_m; instead, it is not satisfied whenever distributions $F(m, t)$ are power-law, that is, $F(m, t) \approx (\alpha_m t)^{-k}$.

4.3. RANDOM and FIFO

The decoupling principle can be easily extended to RANDOM/FIFO caching policies by reinterpreting $T_C(m)$ as the (in general, random) sojourn time of content m in the cache. In the same spirit of the original Che's approximation, we assume $T_C(m) = T_C$ to be a primitive *random* variable (not any more a constant) whose distribution does not depend on m.

Under IRM traffic, the dynamics of each content m in the cache can be described by an M/G/1/0 queuing model. Observe that object m, when not in the cache, enters it according to a Poisson arrival process. Then, it stays in the cache for a duration equal to T_C, after which it is evicted *independently* of the arrival of other requests for content m during the sojourn time.

The expression of $p_{\text{in}}(m)$ and $p_{\text{hit}}(m)$ can then be immediately obtained from the Erlang-B formula (exploiting PASTA):

$$p_{\text{hit}}(m) = p_{\text{in}}(m) = \lambda_m \mathbb{E}[T_C]/(1 + \lambda_m \mathbb{E}[T_C]).$$

Note that we still employ Equation (2) to compute $\mathbb{E}[T_C]$.

As an immediate consequence of Erlang-B insensitivity property to the distribution of service time, we conclude the following.

PROPOSITION 4.1. *Under IRM traffic, the performance of RANDOM and FIFO (in terms of hit probability) are the same.*

This result was originally obtained formally by Gelenbe [1973] using a totally different approach that does not resort to Che's approximation.

Note that, under FIFO policy, we can assume T_C to be a constant, in perfect analogy to LRU. T_C is still equal to the time needed to observe the requests for C distinct objects arriving at the cache. On the other hand, under RANDOM policy, it is natural to approximate the sojourn time of an object in the cache with an exponential distribution. Under RANDOM, an object is evicted with probability $1/C$ upon arrival of each request for an object that is not in the cache.

Under *renewal* traffic, the dynamics of each object under FIFO and RANDOM can be described, respectively, by a G/D/1/0 and a G/M/1/0 queuing model. Observe that, under general traffic, the performance of FIFO and RANDOM are not necessarily the same.

We now show how the RANDOM policy can be analyzed, under *renewal traffic*, employing basic queuing theory. Probability p_{hit} can be obtained as the loss probability of the G/M/1/0 queue. Simply put, the hit probability $p_{\text{hit}}(m)$ of a given content m equals the probability that the content has not been evicted before the arrival of the next request for content m. Having approximated the sojourn time in the cache by an exponential distribution, we can easily compute the following:

$$p_{\text{hit}}(m) = \int_0^\infty e^{-r/\mathbb{E}[T_C]} \, \mathrm{d}F_R(r) = M_R(m, -1/\mathbb{E}[T_C]),$$

where $M_R(m, \cdot)$ is the moment generating function of object-m interrequest time.

Probability $p_{\text{in}}(m)$ can also be obtained exploiting the fact that the dynamics of a G/M/1/0 system are described by a process that regenerates at each arrival. On this process, we can perform a standard cycle analysis as follows (we drop the dependency of random variables on m to simplify the notation). We denote by T_{cycle} the duration of a cycle (which corresponds to an interrequest interval). Observe that, by construction, the object is surely in the cache at the beginning of a cycle. Let τ be the residual time spent by the object in the cache, since a cycle has started, and T_{ON} be the time spent by the object in the cache within a cycle.

By definition, $T_{\text{ON}} = \min\{\tau, T_{\text{cycle}}\}$. Thus, by standard renewal theory, we have $p_{\text{in}}(m) = \mathbb{E}[T_{\text{ON}}]/\mathbb{E}[T_{\text{cycle}}]$. Figure 2 illustrates the two cases that can occur, depending on whether the object is evicted or not before the arrival of the next request. Now, we know that $\mathbb{E}[T_{\text{cycle}}] = 1/\lambda_m$. For $\mathbb{E}[T_{\text{ON}}]$, we obtain the following:

$$\begin{aligned}\mathbb{E}[T_{\text{ON}}] &= \int_0^\infty \left(\mathbb{E}[T_{\text{ON}} \cdot \mathbb{I}_{\tau \le r} \mid T_{\text{cycle}} = r] + \mathbb{E}[T_{\text{ON}} \cdot \mathbb{I}_{\tau > r} \mid T_{\text{cycle}} = r]\right) \mathrm{d}F_R(r) \\ &= \int_0^\infty \left(\int_0^r \frac{x}{\mathbb{E}[T_C]} e^{-x/\mathbb{E}[T_C]} \, \mathrm{d}x + r e^{-r/\mathbb{E}[T_C]} \right) \mathrm{d}F_R(r). \end{aligned} \tag{8}$$

In the end, we get that $p_{\text{in}}(m) = \lambda_m \, \mathbb{E}[T_C] \, (1 - M_R(m, -1/\mathbb{E}[T_C]))$.

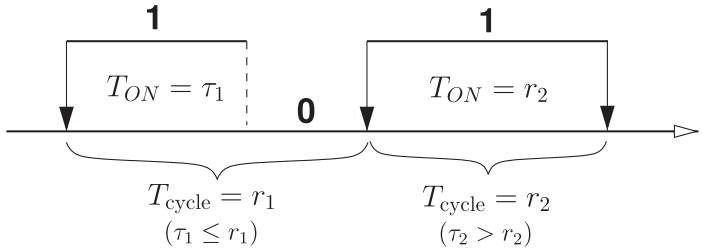

Fig. 2. Illustration of the cycle analysis used for deriving Equation (8). Vertical arrows represent incoming requests for the content.

4.4. 2-LRU

We now move to the k-LRU strategy, considering first the simple case of $k = 2$. For this system, we derive both a rough approximation based on an additional simplifying assumption (which is later used to analyze the more general k-LRU) and a more refined model that is based only on Che's approximation. For both models, we consider either IRM or *renewal* traffic.

Let T_C^i be the eviction time of cache i. We start observing that meta-cache 1 behaves exactly like a standard LRU cache, for which we can use previously derived expressions. Under IRM, $p_{\text{in}}(m)$ and $p_{\text{hit}}(m)$ (which are identical by PASTA) can be approximately derived by the following argument: object m is found in cache 2 at time t if and only if the last request arrived in $\tau \in [t - T_C^2, t)$ and either object m was already in cache 2 at time τ^- or it was not in cache 2 at time τ^-, but its hash was already stored in meta-cache 1. Under the additional approximation that the states of meta-cache 1 and cache 2 are independent at time τ^-, we obtain the following:

$$p_{\text{hit}}(m) = p_{\text{in}}(m) \approx \left(1 - e^{-\lambda_m T_C^2)}\right)\left[p_{\text{hit}}(m) + \left(1 - e^{-\lambda_m T_C^1}\right)(1 - p_{\text{hit}}(m))\right]. \tag{9}$$

Observe that the independence assumption between cache 2 and meta-cache 1 is reasonable under the assumption that T_C^2 is significantly larger than T_C^1 (which is typically the case when the two caches have the same size). In this case, the states of cache 2 and meta-cache 1 tend to desynchronize, since a hash is expunged by meta-cache 1 before the corresponding object is evicted by cache 2, making it possible to find an object in cache 2 and not in meta-cache 1 (which, otherwise, would not be possible if $T_C^1 \geq T_C^2$).

An exact expression for $p_{\text{hit}}(m)$ (under Che's approximation) that does not require any independence assumption can be derived observing that the dynamics of object m in the system, sampled at request arrivals, can be described by the four-states Discrete Time Markov Chain (DTMC) represented in Figure 3, in which each state is denoted by a pair of binary variables indicating the presence of object m in meta-cache 1 and cache 2, respectively. Solving the DTMC, we get the following:

$$p_{\text{hit}}(m) = p_{\text{in}}(m) = 1 - \frac{(1 + q_a)q_b}{q_a + q_b}, \tag{10}$$

with $q_a = 1 - e^{-\lambda_m T_C^1}$, $q_b = e^{-\lambda_m T_C^2}$ and $q_c = 1 - (q_a + q_b)$.

The extension to *renewal* traffic can be carried out following the same lines as before. Under the additional independence assumption between the two caches, we obtain the following:

$$\begin{aligned} p_{\text{hit}}(m) &\approx F_R\left(m, T_C^2\right)\left[p_{\text{hit}}(m) + F_R\left(m, T_C^1\right)(1 - p_{\text{hit}}(m))\right] \\ p_{\text{in}}(m) &\approx \hat{F}_R\left(m, T_C^2\right)\left[p_{\text{hit}}(m) + F_R\left(m, T_C^1\right)(1 - p_{\text{hit}}(m))\right]. \end{aligned}$$

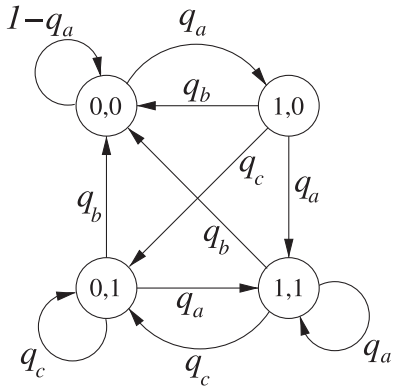

Fig. 3. DTMC describing the dynamics of an object in 2-LRU, sampled at request arrival times.

Also, the refined model can be generalized to *renewal* traffic, observing that object-m dynamics in the system, sampled at request arrivals (i.e., logically just before a request arrival), are still described by a Markov Chain with exactly the same structure as in Figure 3 (only the expressions of transition probabilities change in an obvious way). Thus, we obtain the following:

$$p_{\text{hit}}(m) = 1 - \frac{(1+q_a)q_b}{q_a+q_b},$$

with $q_a = F(m, T_C^1)$ and $q_b = 1 - F(m, T_C^2)$.

To compute $p_{\text{in}}(m)$, we can resort to a cycle analysis, whose details are reported in Appendix C.

4.5. k-LRU

Previous expressions obtained for 2-LRU (under the independence assumption between caches) can be used to iteratively compute the hit probabilities of all caches in a k-LRU system. For example, under IRM, we can use Equation (9) to relate the hit probability of object m in cache i, $p_{\text{hit}}(i, m)$, to the hit probability $p_{\text{hit}}(i-1, m)$ of object m in the previous cache, obtaining the following:

$$p_{\text{hit}}(i,m) = p_{\text{in}}(i,m) \approx \left(1 - e^{-\lambda_m T_C^i}\right)[p_{\text{hit}}(i,m) + (p_{\text{hit}}(i-1,m))(1 - p_{\text{hit}}(i,m))]. \quad (11)$$

The generalization to *renewal* traffic is straightforward.

At last, for large k, we can state:

THEOREM 4.2. *According to* (11) *k-LRU tends asymptotically to LFU as* $k \to \infty$ *under IRM and renewal traffic, as long as the support of the inter-request time distribution is unbounded and for any* m_1 *and* m_2*, with* $\lambda_{m_1} < \lambda_{m_2}$*, it holds* $\lim_{t\to\infty} \frac{1-F(m_1,t)}{1-F(m_2,t)} > 1$.

The proof is reported in Appendix D.

4.6. k-RANDOM

k-RANDOM can be analyzed under Che's approximation assuming exponential sojourn times in the caches. As an example, the dynamics of an object in 2-RANDOM (under IRM traffic) are described by the simple four-states, continuous time Markov chain depicted in Figure 4. In general, k-RANDOM can be exactly analyzed by solving a continuous time Markov chain with 2^k states. We omit the details of such standard analysis here.

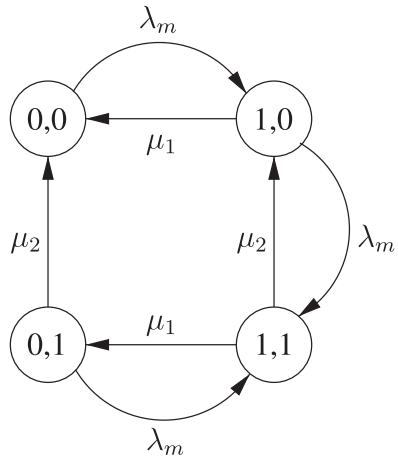

Fig. 4. CTMC describing the dynamics of an object in 2-RANDOM. We denoted $\mu_1 = 1/T_C^1$, $\mu_2 = 1/T_C^2$.

4.7. Small-Cache Approximations

Small-cache approximations can be obtained by replacing the expressions of $p_{\text{hit}}(m)$ and $p_{\text{in}}(m)$ with their truncated Taylor expansion (with respect to $T_C \to 0$). This is especially useful to understand the dependency of p_{in} and p_{hit} on the object arrival rate λ_m (thus its popularity), obtaining interesting insights into the performance of the various caching policies. We restrict ourselves to IRM traffic; however, we emphasize that a similar approach can be generalized to *renewal* traffic. We obtain the following:

$$p_{\text{hit}}(m) = p_{\text{in}}(m) \approx \begin{cases} \lambda_m T_C - \frac{(\lambda_m T_C)^2}{2} & \text{LRU} \\ \lambda_m T_C - (\lambda_m T_C)^2 & \text{RANDOM/FIFO} \\ q\lambda_m T_C + q(\frac{1}{2} - q)(\lambda_m T_C)^2 & \text{q-LRU} \\ (\lambda_m)^k \prod_{i=1}^{k} T_C^i & \text{k-LRU} \end{cases}$$

Previous expressions permit us immediately to rank the performance of the considered policies in the small-cache regime. Specifically, better performance is achieved by caching policy under which $p_{\text{hit}}(m)$ exhibits stronger dependency on λ_m. Recall that (under IRM) $p_{\text{hit}} = \sum_m \frac{\lambda_m}{\Lambda} p_{\text{hit}}(m)$, while $\sum_m p_{\text{hit}}(m) = \sum_m p_{\text{in}}(m) = C$. Hence, the stronger the dependency of $p_{\text{hit}}(m)$ on λ_m, the more closely a policy tends to approximate the behavior of LFU (the optimal policy), which statically places in the cache the C top popular contents.

Therefore, k-LRU turns out to be the best strategy, since the dependency between $p_{\text{hit}}(m)$ and content popularity λ_m is polynomial of order $k \geq 2$, in contrast to other policies (including q-LRU for fixed q) for which $p_{\text{hit}}(m)$ depends linearly on λ_m. The coefficient of the quadratic term further allows us to rank policies other than k-LRU: q-LRU is the only policy exhibiting a positive quadratic term (for small q), which makes the dependency of $p_{\text{hit}}(m)$ on λ_m slightly superlinear. At last, LRU slightly outperforms RANDOM/FIFO because its negative quadratic term has a smaller coefficient.

4.8. Model Validation and Insights

The goal of this section is twofold. First, we wish to validate previously derived analytical expressions against simulations, showing the surprising accuracy of our approximate models in all considered cases. Second, we evaluate the impact of system/traffic parameters on cache performance, obtaining important insights for network design.

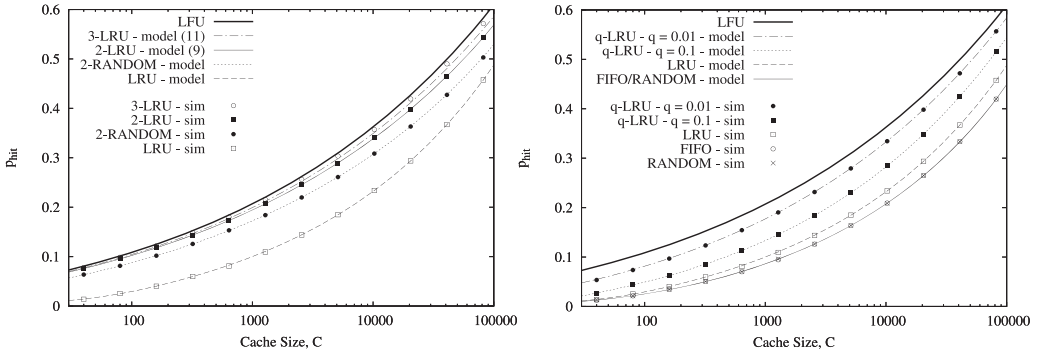

Fig. 5. Hit probability versus cache size for various caching policies under IRM.

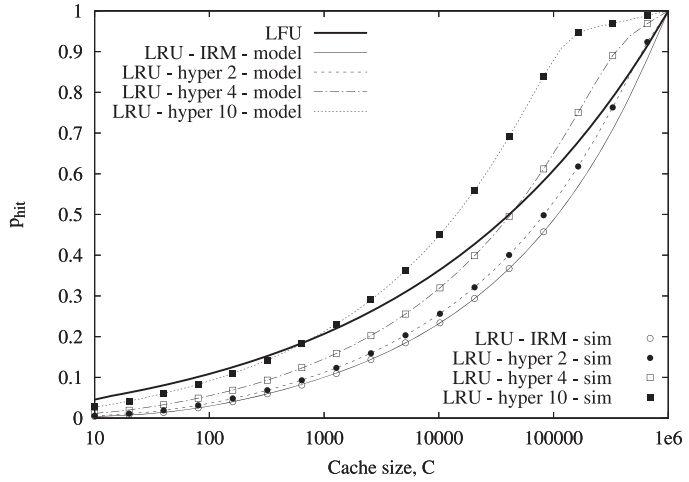

Fig. 6. Hit probability vs cache size, for LRU, under different degrees of temporal locality.

Unless otherwise specified, we will always consider a catalogue size of $M = 10^6$ and a Zipf's law exponent $\alpha = 0.8$.

Figure 5 reports the hit probability achieved by the different caching strategies that we have considered under IRM traffic. Analytical predictions are barely distinguishable from simulation results, also for the 3-LRU system, for which our approximation (Equation (11)) relies on an additional independence assumption among the caches.

As theoretically predicted, q-LRU (k-LRU) approaches LFU as $q \to 0$ ($k \to \infty$). Interestingly, the introduction of a single meta-cache in front of an LRU cache (2-LRU) provides huge benefits, getting very close to optimal performance (LFU).

Differences among the hit probability achieved by the various caching policies become more significant in the small-cache regime (spanning almost 1 order of magnitude). In this case, insertion policies providing some protection against unpopular objects largely outperform policies that do not filter any request. Instead, the impact of the eviction policy appears to be much weaker, with LRU providing moderately better performance than RANDOM/FIFO.

Figure 6 shows the impact of temporal locality on caching performance: LRU is evaluated under *renewal* traffic in which object interarrival times are distributed according to a second-order hyperexponential with branches $\lambda_m^1 = z\lambda_m$ and $\lambda_m^2 = \lambda_m/z$ (hereinafter,

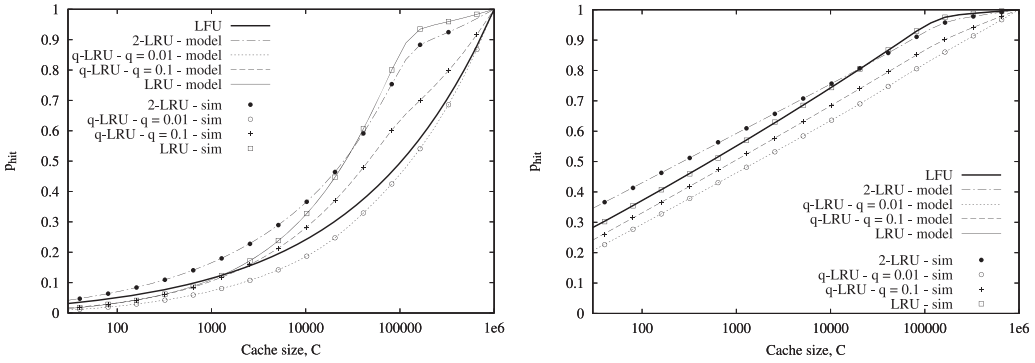

Fig. 7. Hit probability versus cache size, for various caching policies, under hyper-10 traffic, in the case of $\alpha = 0.7$ (left plot) or $\alpha = 1$ (right plot).

we will call hyper-z such distribution), so that increasing values of z results in stronger temporal locality in the request process. We observe that temporal locality can have a dramatic (beneficial) impact on hit probability; hence, it is crucial to take it into account while developing analytical models of cache performance.

Figure 6 also shows that LFU is no longer optimal when traffic does not satisfy the IRM. This is because LFU statically places in the cache the C most popular objects (on the basis of the *average* request rate of contents). Hence, the content of the cache is never adapted to instantaneous traffic conditions, resulting in suboptimal performance.

Figure 7 compares the performance of LFU, LRU, q-LRU and 2-LRU in the case in which traffic exhibits significant temporal locality (hyper-10). We also change the Zipf's law exponent, considering either $\alpha = 0.7$ (left plot) or $\alpha = 1.0$ (right plot).

We observe that q-LRU performs poorly in this case, especially for small values of q (in sharp contrast to what we have seen under IRM). This is because q-LRU with very small q tends to behave like LFU (keeping statically in the cache only the objects with the largest *average* arrival rate), which turns out to be suboptimal as it does not benefit from the temporal locality in the request process.

On the contrary, a simple 2-LRU system also provides very good performance in the presence of strong temporal locality. This is because, while 2-LRU is able to filter out unpopular contents, its insertion policy is fast enough to locally adapt to short-term popularity variations induced by temporal locality.

To further validate the design insights gained by our analysis, we have run a trace-driven experiment, using a real trace of YouTube video requests collected inside the network of a large Italian ISP, offering Internet access to residential customers. The trace has been extracted analyzing TCP flows by means of Tstat, an open-source traffic-monitoring tool developed at Politecnico di Torino [Finamore et al. 2011]. During a period of 35 days in 2012, from March 20th to April 25th, we recorded in total 3.8M requests, for 1.76M videos, coming from 31124 distinct IP addresses.

Figure 8 reports the hit probability achieved by different caching schemes[3]. We observe that most considerations drawn under synthetic traffic (in particular, the policy ranking) still hold when the cache is fed by real traffic taken from an operational network. We summarize the main findings: (i) the insertion policy plays a crucial role in cache performance, especially in the small-cache regime; (ii) a single meta-cache (2-LRU system) significantly outperforms the simple LRU and its probabilistic version

[3]The largest cache size that we could consider was limited by the finite duration of the trace.

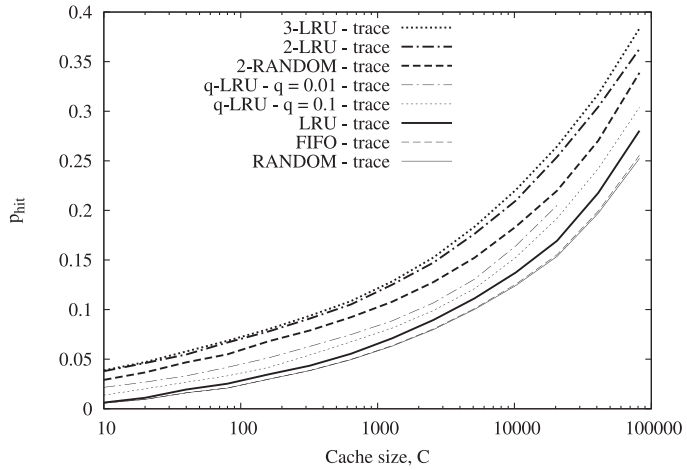

Fig. 8. Hit probability versus cache size for various caching policies under real trace of YouTube video requests.

(q-LRU), while additional meta-caches provide only minor improvements; and (iii) the impact of the eviction policy is not significant, especially when caches are small with respect to the catalogue size.

5. CACHE NETWORKS

In a typical cache network, caches forward their miss stream (i.e., requests that have not found the target object) to other caches. Let us briefly recall the standard approach that has been proposed in the literature to analyze this kind of system.

We first introduce some notation. Let $p_{\text{hit}}(i, m)$ be the hit probability of object m in cache i, and $p_{\text{in}}(i, m)$ be the (time average) probability that object m is in cache i. We denote by T_C^i the eviction time of cache i. Furthermore, let $\overline{\lambda}_m(i)$ be the total *average* arrival rate of requests for object m at cache i. This rate can be immediately computed, provided that we know the hit probability of object m at all caches sending their miss stream to cache i (see Equation (14)).

Once we know the average arrival rates $\overline{\lambda}_m(i)$, we can simply assume that the arrival process of requests for each object at any cache is Poisson, and thus independently solve each cache using its IRM model. A multivariable, fixed-point approach is then used to solve the entire system (see Rosensweig et al. [2010] for a dissection of the errors introduced by this technique).

We now explain how Che's approximation can be exploited to obtain a more accurate analysis of the cache network under the three replication strategies defined in Section 2.4. To describe our improved technique, it is sufficient to consider the simple case of just two caches (tandem network). The extension of our method to general networks is straightforward.

We will limit ourselves to the case of networks of LRU caches in which the traffic produced by the users satisfies the IRM model (i.e., the exogenous process of requests for each object is Poisson). The general idea is to try to capture (though still in an approximate way) the existing correlation among the states of neighboring caches, which is totally neglected under the Poisson approximation. To do so, a different approximation is needed for each considered replication strategy, as explained in the following sections.

5.1. Leave-Copy-Everywhere

Focusing on the basic case of a tandem network, the arrival process of requests for object m at the first cache is an exogenous Poisson process of rate $\lambda_m(1)$. The first cache (which is not influenced by the second one) can then be solved using the standard IRM model, giving

$$p_{\text{hit}}(1, m) = p_{\text{in}}(1, m) = 1 - e^{-\lambda_m(1)T_C^1}. \tag{12}$$

The arrival process of a request for object m at the second cache is not Poisson. It is, instead, an ON-OFF modulated Poisson process, in which the ON state corresponds to the situation in which object m is not stored in cache 1, so that requests for this object are forwarded to cache 2. Instead, no requests for object m can arrive at cache 2 when m is present at cache 1 (OFF state).

The standard approximation would be to compute the average arrival rate $\overline{\lambda}_m(2) = \lambda_m(1)(1 - p_{\text{hit}}(1, m))$ and to apply the IRM model also to the second cache. Can we do better than this? Actually, yes, at least to compute the hit probability $p_{\text{hit}}(2, m)$, which can, in practice, be very different from $p_{\text{in}}(2, m)$ since PASTA does not apply.

We observe that a request for m can arrive at time t at cache 2 only if object m is not stored in cache 1 at t^-. This implies that no exogenous requests can have arrived in the interval $[t - T_C^1, t]$ (otherwise, m would be present in cache 1 at time t); hence, *a fortiori*, no requests for m can have arrived at cache 2 in the same interval.

Now, provided that $T_C^2 > T_C^1$, object m is found in cache 2 at time t if and only if at least one request arrived at cache 2 within the interval $[t - T_C^2, t - T_C^1]$. During this interval, the arrival process at cache 2 is not Poisson (it depends on the unknown state of cache 1), and we resort to approximating it by a Poisson process with rate $\overline{\lambda}_m(2)$, obtaining the following:

$$p_{\text{hit}}(2, m) \approx 1 - e^{-\overline{\lambda}_m(2)(T_C^2 - T_C^1)}. \tag{13}$$

Essentially, the improvement with respect to the standard approximation consists of the term $T_C^2 - T_C^1$ in Equation (13), in place of T_C^2. If, instead, $T_C^2 < T_C^1$, we clearly have $p_{\text{hit}}(2, m) = 0$.

Note that this reasoning cannot be applied to compute $p_{\text{in}}(2, m)$ (which is necessary to estimate T_C^2). Thus, we simply express

$$p_{\text{in}}(2, m) \approx 1 - e^{-\overline{\lambda}_m(2)T_C^2},$$

as in the standard IRM model.

To show the significant gains in terms of accuracy that can be obtained by applying our simple improved approximation with respect to the Poisson approximation, we consider a tandem network in which the first cache is fed by IRM traffic with catalogue size $M = 10^6$ and Zipf's law exponent $\alpha = 0.8$. Figure 9 reports both the total hit probability and the hit probability on the second cache, under the two considered approximations, against simulation results. We observe that the Poisson approximation tends to overestimate the total hit probability, essentially as a consequence of a large overestimate of the hit probability on the second cache. Our improved approximation, which, recall, essentially leads to substituting T_C^2 with $T_C^2 - T_C^1$ in the standard formula to compute the hit probability of the second cache, brings back the analytical prediction of total hit probability very close to simulation results, thanks to a much better model of the behavior of the second cache.

5.2. Leave-Copy-Probabilistically

Also in this case, the first cache is not influenced by the second; hence, we can use the IRM formula of q-LRU (Equation (5)) to analyze its behavior.

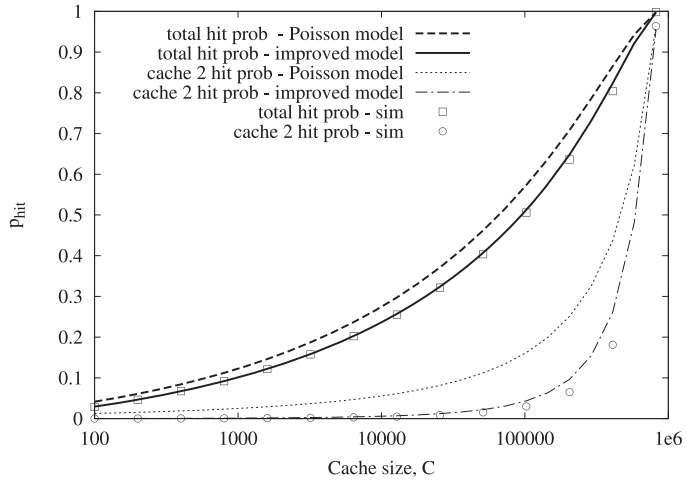

Fig. 9. Comparison between Poisson approximation and our improved approximation in the case of a tandem network of two LRU caches under IRM traffic.

To evaluate $p_{\text{hit}}(2, m)$, we observe that a request for content m that arrives at time t at cache 2 produces a hit if and only if at time t^- content m is stored at cache 2 but not in cache 1. For this to happen, in the case that $T_C^2 > T_C^1$, there are two sufficient and necessary conditions related to the *previous* request for m arriving at cache 2: (i) this request produced a hit at cache 2, or it triggered an insertion here; and (ii) it arrived at cache 2 either in the interval $[t - T_C^2, t - T_C^1]$, or in the interval $[t - T_C^1, t]$ without triggering an insertion in cache 1. We remark that, in contrast to the LCE case, now it is possible that the previous request arrived in the interval $[t - T_C^1, t]$: the previous request can arrive in this interval, produce a miss in cache 1 (thus be forwarded to cache 2) and *not* trigger an insertion in cache 1, so that we can really observe another request arriving at cache 2 at time t. To evaluate the probability of this event, we model the stream of requests arriving at cache 2 (i.e., producing a miss at cache 1) without triggering an insertion in cache 1 as a Poisson process with intensity $\overline{\lambda}_m(2) \cdot (1 - q)$. We obtain the following:

$$p_{\text{hit}}(2, m) \approx [p_{\text{hit}}(2, m) + q(1 - p_{\text{hit}}(2, m))] \cdot \left(1 - e^{-\overline{\lambda}_m(2)(T_C^2 - T_C^1)} \cdot e^{-\overline{\lambda}_m(2)(1-q)T_C^1}\right).$$

In this expression, the first term of the product refers to condition (i), whereas the second term accounts for condition (ii) going through the complementary event that no requests arrive at cache 2 either in the interval $[t - T_C^1, t]$ or in the interval $[t - T_C^2, t - T_C^1]$. Note that this expression reduces to Equation (13) when $q = 1$ (i.e., LCE).

If, instead, $T_C^2 < T_C^1$, the formula simplifies to

$$p_{\text{hit}}(2, m) \approx [p_{\text{hit}}(2, m) + q(1 - p_{\text{hit}}(2, m))]\left(1 - e^{-\overline{\lambda}_m(2)(1-q)T_C^2}\right).$$

To estimate $p_{\text{in}}(2, m)$, we resort to the standard Poisson approximation:

$$p_{\text{in}}(2, m) \approx \left(1 - e^{-\overline{\lambda}_m(2)T_C^2}\right)[p_{\text{in}}(2, m) + q(1 - p_{\text{in}}(2, m))].$$

5.3. Leave-Copy-Down

This strategy is more complex to analyze, since now the dynamics of cache 1 and cache 2 depend mutually on each other. It is possible to insert a content in cache 1 only when it is already stored in cache 2. Probability $p_{\text{in}}(1, m)$ can be computed considering that object m is found in cache 1 if and only if the last request arrived in $[t - T_C^1, t]$ and either (i) it hit the object in cache 1 or (ii) it found the object in cache 2 (and not in cache 1).

Since PASTA holds, we have that

$$p_{\text{in}}(1,m) \approx p_{\text{hit}}(1,m) = [(1-p_{\text{in}}(1,m))\, p_{\text{hit}}(2,m) + p_{\text{in}}(1,m)] \cdot \left(1 - e^{-\lambda_m(1)T_C(1)}\right).$$

Observe in the previous expression that we have assumed the states of cache 1 and cache 2 to be independent. On the other hand, similar to what we have done before, we write

$$p_{\text{in}}(2,m) \approx \left(1 - e^{-\overline{\lambda}_m(2)T_C^2}\right).$$

Note that, since $p_{\text{in}}(1,m)$ and $p_{\text{in}}(2,m)$ are interdependent, a fixed-point iterative procedure is needed to jointly determine them.

It remains to approximate the hit probability at cache 2. When $T_C^2 > T_C^1$, we write

$$p_{\text{hit}}(2,m) \approx \left(1 - e^{-\overline{\lambda}_m(2)(T_C^2 - T_C^1)}\right) e^{-\overline{\lambda}_m(2)T_C^1} + \left(1 - e^{-\overline{\lambda}_m(2)(1-p_{\text{hit}}(2,m))T_C^1}\right).$$

Since at time t^-, cache 1 does not store the object by construction, either the previous request arrived in $[t - T_C^2, t - T_C^1]$ at cache 2 or it arrived in $[t - T_C^1, t]$ (again at cache 2), but it did not trigger an insertion in cache 1 because object m was not found in cache 2. As before, we model the stream of requests arriving at cache 2 (i.e., producing a miss at cache 1) without triggering an insertion in the first cache as a Poisson process with intensity $\overline{\lambda}_m(2) \cdot (1 - p_{\text{hit}}(2,m))$.

Similarly, if $T_C^2 < T_C^1$, then

$$p_{\text{hit}}(2,m) \approx \left(1 - e^{-\lambda_m(2)(1-p_{\text{hit}}(2,m))T_C^2}\right).$$

5.4. Extension to General Cache Networks

Our approach, which has been described earlier for the simple case of a tandem network, can be easily generalized to any network. We limit ourselves to explaining how this can be done for the LCE scheme. Let $r_{j,i}$ be the fraction of requests for object m that are forwarded from cache j to cache i (in the case of a miss in cache j). Observe that $[r_{j,i}]$ depends on the routing strategy of requests adopted in the network, and can be considered as a given input to the model.

The average arrival rate of requests for m at i is then

$$\overline{\lambda}_m(i) = \sum_j \overline{\lambda}_m(j)(1 - p_{\text{hit}}(j,m))r_{j,i}, \tag{14}$$

and we can immediately express the following:

$$p_{\text{in}}(i,m) \approx 1 - e^{-\overline{\lambda}_m(i)T_C^i},$$

resorting to the standard Poisson approximation.

Our refined approach to estimating the hit probability can still be applied to the computation of the conditional probability $p_{\text{hit}}(i,m \mid j)$, which is the probability that a request for object m hits the object at cache i, given that it has been forwarded by cache j. This event occurs if and only if either a request arrived at i from j in the time interval $[t - T_C^i, t - T_C^j]$ (provided that $T_C^i > T_C^j$) or at least one request arrived at i in the interval $[t - T_C^i, t]$ from another cache (different from j). Thus, we write the following:

$$p_{\text{hit}}(i,m \mid j) \approx 1 - e^{-A_{i,j}},$$

where

$$A_{i,j} = r_{j,i}\overline{\lambda}_m(j)(1 - p_{in}(j,m)) \max\left(0, T_C^i - T_C^j\right) + \sum_{k \neq j} r_{k,i}\overline{\lambda}_m(k)(1 - p_{in}(k,m))T_C^i.$$

The expression for $p_{\text{hit}}(i,m)$ can then be obtained deconditioning with respect to j.

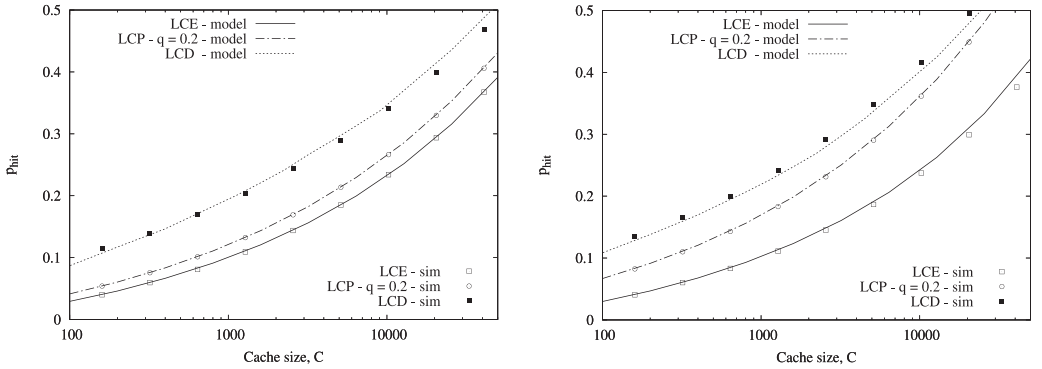

Fig. 10. Hit probability versus cache size for various replication strategies in the case of a chain of 6 caches under IRM traffic. Hit probability of the first cache (left plot) and total hit probability of the network (right plot).

Now, in case of tree-like networks, previous expressions can be evaluated step-by-step starting from the leaves and going up toward the root. In the case of general mesh networks, a global (multivariate) fixed-point procedure is necessary.

5.5. Model Validation and Insights

As before, our aim here is to jointly validate our analytical models against simulation, while getting interesting insights into system behavior.

Figure 10 compares the performance of the different replication strategies that we have analyzed, in the case of a chain of 6 identical caches. We have chosen a chain topology to validate our model, because this topology is known to produce the largest degree of correlation among caches (thus the maximum deviation from the Poisson approximation).

We separately show the hit probability on the first cache (left plot) and the hit probability of the entire cache network (right plot), observing excellent agreement between analysis and simulation in all cases. We note that LCP significantly outperforms LCE, as it better exploits the aggregate storage capacity in the network, avoiding the simultaneous placement of the object in all caches. Yet, LCD replication strategy performs even better, thanks to an improved filtering effect (LCD can be regarded as the dual of k-LRU for cache networks).

Then, we consider a very large topology, comprising 1365 caches, corresponding to a 4-ary regular tree with 6 levels. This topology is extremely expensive (if not impossible) to simulate, whereas the model can predict its behavior at the same computation cost of previous chain topology. Figure 11 reports the total hit probability achieved in this large network for two traffic scenarios (analytical results only).

We again observe the huge gain of LCD with respect to LCE, whereas the benefits of LCP are not very significant, especially with $\alpha = 0.7$.

At last, we consider an example of mesh network comprising 9 caches arranged on a ring topology. Requests can enter the ring at any point, that is, any of the caches along the ring acts as an ingress cache. Requests are forwarded clockwise along the ring. However, requests that have traversed 4 caches without hitting the content are redirected to a remote common repository storing all contents. Figure 12 shows the path followed by the requests arriving externally at one particular cache (requests entering the network at the other caches are treated in a similar way). The total external traffic of incoming requests is uniformly distributed over the 9 caches.

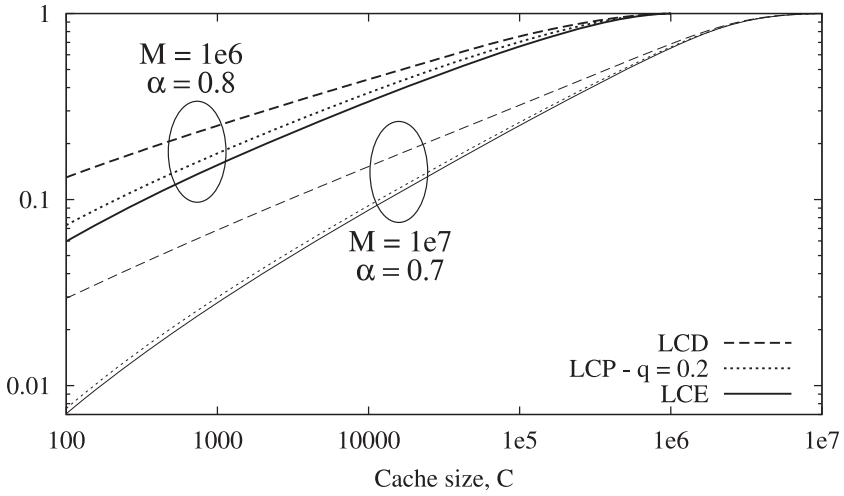

Fig. 11. Hit probability versus cache size for various replication strategies in the case of a tree topology with 1365 caches for two traffic scenarios.

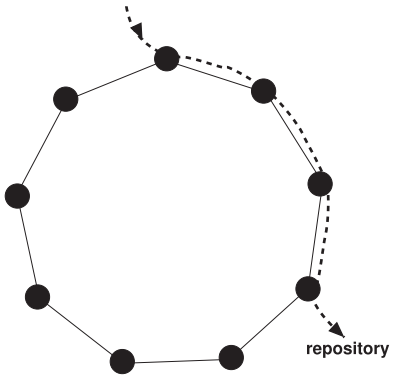

Fig. 12. Ring topology of 9 caches. The path followed by requests entering one particular cache is shown as a dashed line.

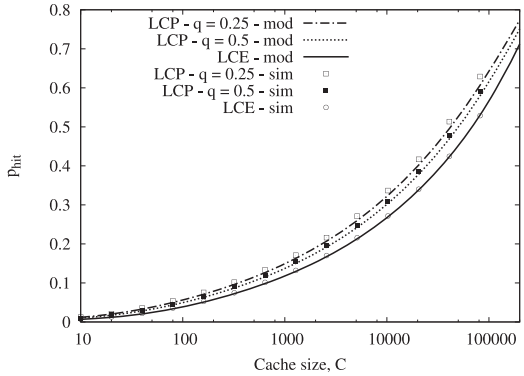

Fig. 13. Performance of LCE and LCP (with $q = 0.5$ or $q = 0.25$) in the ring topology. Comparison between analysis and simulation.

Figure 13 compares the performance of LCE and LCP (with either $q = 0.5$ or $q = 0.25$) in the considered mesh network, showing the global hit probability achieved by the caching system. Here, we have chosen the usual setting of $M = 10^6$ and $\alpha = 0.8$. We have not considered in this scenario the LCD replication strategy, which is primarily meant for hierarchical (tree-like) caching systems and whose performance on general networks with cyclic topology are typically worse than LCP [Rossini and Rossi 2014].

Observe that, also in the more challenging case of cache networks including cycles, the application of our model leads to significantly accurate predictions of the hit-probability. We wish to recall that networks that do not have feed-forward topology cannot be analyzed with existing techniques, such as that proposed in Fofack et al. [2014b].

6. RELATED WORK

The literature on caching systems is vast; thus, we limit ourselves to mentioning the papers more closely related to our work, mainly with a modeling focus. The first

attempts to characterize the performance of simple caching systems date back to the early 1970s [King 1971; Gelenbe 1973]. King [1971] has shown that the computational complexity of an exact model of a single LRU or FIFO cache grows exponentially with both the cache size C and the catalogue size M. In Gelenbe [1973], it was proven that FIFO and RANDOM replacement policies achieve exactly the same hit probability under IRM traffic. Given that an exact characterization of most caching policies is prohibitive, approximated methodologies for the analysis of these systems have been proposed over the years [Dan and Towsley 1990; Che et al. 2002]. Dan and Towsley [1990] propose an approximate technique with complexity $O(CM)$ for the estimation of the hit probability in an LRU cache under IRM. This technique can be extended to FIFO caches as well, although, in this case, the asymptotic complexity cannot be precisely determined due to the iterative nature of the model solution. A different approximation for LRU caches under IRM was originally proposed by Che et al. [2002]. This approximation constitutes the starting point of our work, and is explained in detail in Section 3.

Another thread of works (Jelenković [1999], Jelenković and Radovanović [2003, 2004], and Jelenković and Squillante [2006]) has focused on the asymptotic characterization of the hit probability in LRU caches when the catalog size and cache size jointly scale to infinite. In particular, Jelenković [1999] provides a closed-form expression for the asymptotic hit probability in a large LRU cache under IRM traffic with Zipf's exponent $\alpha > 1$. Later works [Jelenković and Radovanović 2003, 2004] have shown that LRU, in the asymptotic regime, exhibits an insensitivity property to traffic temporal locality. Jelenković and Squillante [2006] established the precise conditions on the scaling of parameters under which the insensitivity property holds. More recently, Jelenković and Radovanović [2008] proposed the "persistent-access-caching" (PAC) scheme, showing that it provides nearly optimal asymptotic performance under IRM with Zipf's exponent $\alpha > 1$. We emphasize that the idea behind the PAC scheme shares some similarities with the k-LRU scheme proposed in this work: under both schemes, an insertion policy is added to LRU to prevent unpopular contents from entering the cache. However, the configuration of PAC is harder, as it requires setting several parameters. The k-LRU scheme, instead, is simpler and self-adapting. Other generalizations/extensions of LRU—known as LRU-2Q, LRU-k, and LRFU—have been proposed in Johnson and Shasha [1994], O'Neil et al. [1993], and Lee et al. [2001], respectively. LRU-2Q is essentially equivalent to k-LRU in the case of $k = 2$. Both LRU-k and LRFU, instead, subsume either LRU or LFU by making the choice of the content to be evicted dependent on the pattern of last k observed content requests. k-LRU is somehow complementary to both LRU-k and LRFU, since it enhances only the insertion policy of the classical LRU by restricting access to the cache only to those contents that are sufficiently popular, while preserving the simplicity of LRU eviction.

In the last few years, cache systems have attracted renewed interest in the context of ICN. In Psaras et al. [2011], a Markovian approach has been proposed to approximate the hit probability in LRU caches under IRM. The proposed method, however, is based on Markovian assumptions and cannot be easily extended to non-IRM traffic. In Carofiglio et al. [2011], the approach of Jelenković [1999] has been extended to analyze the chunkization effect on cache performance in an ICN context. An asymptotic characterization (for large caches) of the hit probability achieved by the RANDOM policy is provided in Gallo et al. [2012]. The trade-off between recency and frequency in LRFU has been studied in Li et al. [2012].

Fricker et al. [2012a] provide a theoretical justification to Che's approximation for LRU, and introduce a first attempt to apply Che's approach to non-LRU caches, considering the RANDOM policy under IRM. We emphasize that the approach proposed

in Fricker et al. [2012a] to analyze RANDOM differs substantially from ours, being significantly more complex and hardly extendable to non-IRM traffic. Last, we wish to mention that Che's approximation for LRU has been very recently [Bianchi et al. 2013] extended to non-IRM traffic in special cases, adopting a dual approach with respect to ours.

With respect to all of these works, the goal of our article is different in tha her we show that the decoupling principle underlying Che's approximation is much more general and flexible than originally thought, and can be successfully applied to a broad set of caching policies under different traffic conditions within a unified framework.

For what concerns cache networks, we mention Rosensweig et al. [2013], Rosensweig et al. [2010], and Gallo et al. [2012]. Rosensweig et al. [2013] explores ergodicity conditions for arbitrary (mesh) networks. The models in Rosensweig et al. [2010] and Gallo et al. [2012] rely on the independence assumption among caches, assuming that requests arriving at each cache satisfy the IRM assumptions. In contrast, we propose a methodology to capture the existing correlation among the states of neighboring caches in a computationally efficient manner, considerably improving the accuracy of analytical predictions. Our approach also permits analyzing cache networks adopting tightly coordinated replication strategies such as LCD. Note that cache networks implementing LCD have been previously considered in Laoutaris et al. [2006] for the special case of tandem topologies. Our methodology provides a significantly simpler and higher scalable alternative to the approach devised in Laoutaris et al. [2006], by capturing in a simple yet effective way existing correlations between caches' states, while reducing the number of parameters that must be estimated through fixed-point procedure.

Finally an alternative approach to ours has been recently proposed (Fofack et al. [2014b], Fofack [2014], and Fofack et al. [2014a]) for cache networks with feed-forward topology implementing TTL-based eviction policies. This approach, which can be used to analyze the performance of LRU, RANDOM, and FIFO under Che's approximation, essentially consists of characterizing the interrequest process arriving at noningress caches through a two-step- procedure: (i) the miss stream of (ingress) caches is exactly characterized as a renewal process with given distribution; and (ii) by exploiting known results on the superposition of independent renewal processes, the exact interrequest time distribution at noningress caches is obtained. Observe, however, that the request process at noningress caches is, in general, nonrenewable (since the superposition of independent renewal processes is not guaranteed to be renewable). Thus, while the procedure proposed in Fofack et al. [2014b] is exact for a network of TTL caches with linear topology, it can be applied to a network of caches with tree structure only by approximating the request processes at noningress caches with renewal processes. Recently, a nice refinement of the approach followed by Fofack et al. [2014b] has been proposed in Berger et al. [2014], in which it has shown that the miss stream of TTL-based caches is a Markovian arrival process (MAP), provided that the request process is MAP. In light of the fact that the superposition of independent MAPs is also a MAP, Berger et al. [2014] have derived an exact analytical method for the analysis of feed-forward networks of TTL caches under MAP traffic.

Although the approach in Fofack et al. [2014b] and Berger et al. [2014] is very elegant, and can be potentially extended to renewal traffic, it suffers from the following two limitations: (i) it becomes computationally very intensive when applied to large networks; and (ii) it can be hardly generalized to general mesh networks (nonfeed-forward). Our approach is somehow complementary to the one followed by Fofack et al. [2014b] and Berger et al. [2014] since, while it applies only to IRM traffic, it is much more scalable and readily applicable to networks with general topology.

7. CONCLUSIONS

The main goal of this article was to show that a variety of caching systems (both isolated and interconnected caches), operating under various insertion/eviction policies and traffic conditions, can be accurately analyzed within a unified framework based on a fairly general decoupling principle extending the original Che's approximation. We have also shown that many properties of cache systems can be obtained within our framework in a simple and elegant way, including asymptotic results that would otherwise require significant efforts to be established. From the point of view of system design, our study has revealed the superiority of the k-LRU policy in terms of both simplicity and performance gains. Still many extensions and refinements are possible, especially for cache networks under general traffic.

APPENDIX

A. PROOF OF Q-LRU → LFU: IRM CASE

We first prove that $\lim_{q\to 0} T_C = +\infty$. Consider function $f(T_C, q) \triangleq \sum_m p_{\text{in}}(m)$. From Equation (2), $f(T_C, q) \equiv C$. Recalling Equation (5), we have that

$$f(T_C, q) = \sum_m \frac{q(1 - e^{-\lambda_m T_C})}{e^{-\lambda_m T_C} + q(1 - e^{-\lambda_m T_C})},$$

where previous sum extends over all contents in the catalog (which is assumed to be of finite size M). Deriving this formula, we obtain the following:

$$f_q \triangleq \frac{\partial f}{\partial q} = \sum_m \frac{(1-x)(x+q(1-x)) - q(1-x)^2}{(x+q(1-x))^2}\bigg|_{x=e^{-\lambda_m T_C}} = \sum_m \frac{(1-x)x}{(x+q(1-x))^2}\bigg|_{x=e^{-\lambda_m T_C}} > 0 \tag{15}$$

and

$$\begin{aligned} f_{T_C} &\triangleq \frac{\partial f}{\partial T_C} = \frac{\partial f}{\partial x}\bigg|_{x=e^{-\lambda_m T_C}} \frac{\partial e^{-\lambda_m T_C}}{\partial T_C} = \sum_m \partial\left(\frac{q(1-x)}{x+q(1-x)}\right)/\partial x\bigg|_{x=e^{-\lambda_m T_C}} \left(-\lambda_m e^{-\lambda_m T_C}\right) \\ &= \sum_m \frac{-q(x+q(1-x)) - q(1-x)(1-q)}{(x+q(1-x))^2}\bigg|_{x=e^{-\lambda_m T_C}} \left(-\lambda_m e^{-\lambda_m T_C}\right) \\ &= \sum_m \frac{q\lambda_m e^{-\lambda_m T_C}}{(e^{-\lambda_m T_C} + q(1-e^{-\lambda_m T_C}))^2} > 0. \end{aligned} \tag{16}$$

By the implicit function theorem, we have that

$$\frac{\partial T_C}{\partial q} = \frac{-f_q(T_C, q)}{f_{T_C}(T_C, q)} < 0.$$

We can conclude that T_C is a decreasing function with respect to q; thus, we have that the limit $\lim_{q\to 0} T_C$ exists and equals $\sup_q T_C$. We prove now that this limit is equal to infinity. We define $T_{C,\sup} \triangleq \sup_q T_C(q) = \lim_{q\to 0} T_C$, and we suppose, by contradiction, that this is a finite quantity. In this case, we would have that

$$\lim_{q\to 0} f(T_C, q) = \lim_{q\to 0} \sum_m p_{\text{in}}(m) = \lim_{q\to 0} \sum_m \frac{q(1 - e^{-\lambda_m T_C})}{e^{-\lambda_m T_C} + q(1 - e^{-\lambda_m T_C})} = 0,$$

in contrast with the fact that the previous sum is equal to C, by definition. Thus, $T_{C,\sup} \triangleq \lim_{q\to 0} T_C(q) = +\infty$. We prove now that $T_C(q)$ asymptotically behaves as

$c \log \frac{1}{q}$ for some $c > 0$ as $q \to 0$. We can write the following:

$$\begin{aligned}
&\lim_{q\to 0} \sum_m p_{\text{in}}(m) \\
&= \lim_{q\to 0} \sum_m \frac{q(1-e^{-\lambda_m T_C}))}{e^{-\lambda_m T_C} + q(1-e^{-\lambda_m T_C})} \\
&= \lim_{q\to 0} \sum_m \frac{q+o(q)}{e^{-\lambda_m T_C} + q + o(q)} \\
&= \lim_{q\to 0} \sum_m \frac{1+o(1)}{1+e^{-\lambda_m T_C}/q + o(1)} \\
&= \lim_{q\to 0} \sum_m \frac{1+o(1)}{1+e^{-(\lambda_m T_C - \log(1/q))} + o(1)}
\end{aligned} \tag{17}$$

We note that, if $\frac{T_C}{\log(1/q))}$ becomes arbitrarily large as $q \to 0$, every term in Equation (17) tends to 1, and the sum would be equal to the number of contents, whereas we know that it has to be equal to C. If, on the other hand, $\frac{T_C}{\log(1/q))}$ becomes arbitrarily small as $q \to 0$, every term in the sum in Equation (17) would tend to 0. We can thus conclude that $\frac{T_C}{\log(1/q)}$ is bounded away from both 0 and infinite.

Thus, assuming for the moment that $\lim_{q\to 0} \frac{T_C}{\log(1/q))}$ exists, it must necessarily be equal to $c > 0$. Now, by setting $\lambda^* = 1/c$, we have that

$$\lim_{q\to 0} p_{\text{in}}(m) = \lim_{q\to 0} \frac{1+o(1)}{q^{\frac{\lambda_m}{\lambda^*}-1} + 1 + o(1)} = \begin{cases} 1 & \text{if } \lambda_m \geq \lambda^* \\ 0 & \text{if } \lambda_m < \lambda^*. \end{cases}$$

Note that the previous argument still holds when $\lim_{q\to 0} \frac{T_C}{\log(1/q))}$ does not exist, provided that the following condition is met: (i) no λ_m can be found, with $\lambda^* < \lambda_m \leq \Lambda^*$, such that $0 < \liminf_{q\to 0} \frac{T_C}{\log(1/q))} = \frac{1}{\Lambda^*} < \limsup_{q\to 0} \frac{T_C}{\log(1/q))} = \frac{1}{\lambda_m} < \infty$.

Last, we show, by contradiction, that either $\lim_{q\to 0} \frac{T_C}{\log(1/q))}$ exists or condition (i) is met. Assume that there is an m such that $\lambda^* \leq \lambda_m < \Lambda^*$. Then, denoting with $\mathbb{I}_{\{A\}}$ the indicator function associated to the event $\{A\}$, by construction, it must be both $\sum_m \mathbb{I}_{\{\lambda_m \geq \lambda^*\}} = C$ and $\sum_m \mathbb{I}_{\{\lambda_m \geq \Lambda^*\}} = C$. Thus, $\sum_m \mathbb{I}_{\{\lambda_m \geq \lambda^*\}} = \sum_m \mathbb{I}_{\{\lambda_m \geq \Lambda^*\}}$, which is in contradiction with the assumption.

B. PROOF OF Q-LRU → LFU: GENERAL CASE

To simplify the proof, we assume the support of the interrequest time pdf to be simply connected. Consequently, $F(m, y)$ ($\hat{F}(m, y)$) is a strictly increasing function with respect to variable x (y) on its relevant range, that is, for any x such that $0 < F(m, x) < 1$ ($\forall\, y$ s.t. $0 < \hat{F}(m, y) < 1$). First, we consider the case in which $F(m, x)$ ($\hat{F}(m, y)$) has an infinite support for any m. In this case, we first prove that $\lim_{q\to 0} T_C = +\infty$. Consider function $f(T_C, q) \triangleq \sum_m p_{\text{in}}(m)$. From Equation (2), $f(T_C, q) \equiv C$. Recalling Equations (7) and (6), we have that

$$p_{\text{hit}}(m) = \frac{qF(m, T_C)}{1 - F(m, T_C)(1-q)}$$

and

$$\begin{aligned} p_{\text{in}}(m) &= \hat{F}(m, T_C)[p_{\text{hit}}(m) + q(1 - p_{\text{hit}}(m))] \\ &= \hat{F}(m, T_C)\left[\frac{qF(m, T_C)}{1 - F(m, T_C)(1-q)} + q\left(1 - \frac{qF(m, T_C)}{1 - F(m, T_C)(1-q)}\right)\right] \\ &= \hat{F}(m, T_C)\frac{q}{1 - F(m, T_C)(1-q)}. \end{aligned} \tag{18}$$

Thus,

$$f(T_C, q) = \sum_m \hat{F}(m, T_C)\frac{q}{1 - F(m, T_C)(1-q)}.$$

Deriving this formula, we obtain the following:

$$\begin{aligned} f_q \triangleq \frac{\partial f}{\partial q} &= \sum_m \hat{F}(m, T_C)\frac{1 - F(m, T_C)(1-q) - qF(m, T_C)}{[1 - F(m, T_C)(1-q)]^2} \\ &= \sum_m \hat{F}(m, T_C)\frac{1 - F(m, T_C)}{[1 - F(m, T_C)(1-q)]^2} > 0 \end{aligned}$$

and

$$\begin{aligned} f_{T_C} \triangleq \frac{\partial f}{\partial T_C} &= \sum_m \frac{\partial \hat{F}(m, T_C)}{\partial T_C}\left[\frac{q}{1 - F(m, T_C)(1-q)}\right] + \hat{F}(m, T_C)\frac{\partial}{\partial T_C}\frac{q}{1 - F(m, T_C)(1-q)} \\ &= \sum_m \frac{\partial \hat{F}(m, T_C)}{\partial T_C}\left[\frac{q}{1 - F(m, T_C)(1-q)}\right] \\ &\quad + \hat{F}(m, T_C)\frac{q(1-q)}{[1 - F(m, T_C)(1-q)]^2}\frac{\partial F(m, T_C)}{\partial T_C} > 0, \end{aligned}$$

since both $\hat{F}(m, T_C)$ and $F(m, T_C)$ are increasing with T_C.

By the implicit function theorem, we have that

$$\frac{\partial T_C}{\partial q} = \frac{-f_q(T_C, q)}{f_{T_C}(T_C, q)} < 0.$$

We can conclude that T_C is a decreasing function with respect to q; thus, we have that the limit $\lim_{q\to 0} T_C$ exists and equals $\sup_q T_C$. We now prove that this limit is equal to infinity. We define $T_{C,\sup} \triangleq \sup_q T_C(q) = \lim_{q\to 0} T_C$, and we suppose, by contradiction, that this is a finite quantity. In this case, we would have that

$$\lim_{q\to 0} f(T_C, q) = \lim_{q\to 0} \sum_m p_{\text{in}}(m) = \lim_{q\to 0} \sum_m \hat{F}(m, T_C)\frac{q}{1 - F(m, T_C)(1-q)} = 0,$$

in contrast with the fact that the previous sum is equal to C, by definition. Thus, since $\lim_{q\to 0} T_C(q) = +\infty$ we have that

$$\lim_{q\to 0} p_{\text{in}}(m) = \lim_{q\to 0} \sum_m \hat{F}(m, T_C)\frac{1}{[1 - F(m, T_C)(1-q)]/q}. \tag{19}$$

Now, observe that, if $1 - F(m, T_C) = o(q)$, the previous limit becomes equal to 1, whereas, if $1 - F(m, T_C) = \omega(q)$, the limit is equal to 0.

Then, with similar arguments as for the exponential case, under our assumptions (i.e., the fact that we assume $\lim_{t\to\infty} \frac{1-F(m_1,t)}{1-F(m_2,t)} = \infty$ whenever $\lambda_{m_1} < \lambda_{m_2}$), we can easily

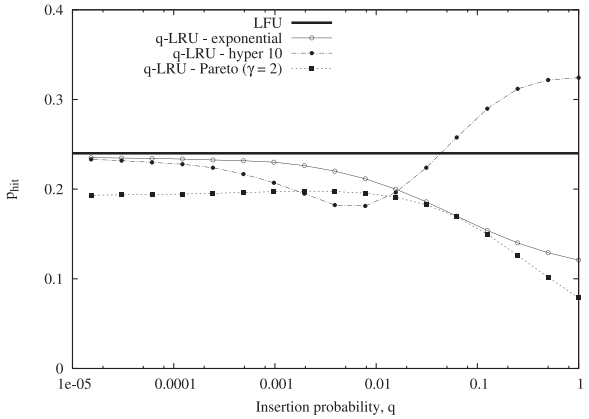

Fig. 14. Hit probability versus insertion probability of q-LRU, for different interrequest time distributions, fixed cache size equal to 10,000, $\alpha = 0.7$.

show that there necessarily exists some m_0 such that $1-F(m, T_C) = o(q)$ for $m < m_0$ and $1 - F(m, T_C) = \omega(q)$ for $m > m_0$. Observe that, by hypothesis, the asymptotic behavior of $1 - F(m, T_C)$ as $T_C \to \infty$ depends on m (i.e., on arrival rates λ_ms, which are assumed to be different for different m).

Figure 14 provides a numerical confirmation of our theoretical predictions (see also Remark after Theorem 4.1), plotting the hit probability as a function of the insertion probability of q-LRU under different interrequest time distributions: exponential, hyper-10, Pareto (with exponent $\gamma = 2$). This experiment suggests that both the exponential and hyper-10 curves approach LFU as $q \to 0$, while the curve corresponding to the Pareto case tends to a different limit.

The case in which $F(m, T_C)$ has a bounded support for some m can be treated similarly. However, if the number of contents with finite support exceeds C, T_C does not tend anymore to ∞. Observe that, from Equation (19), we can deduce that every content whose interrequest time has a maximum value, which is smaller that $T_{C,\text{sup}}$, will be necessarily found in the cache with a probability tending to 1 when $q \to 0$, while every other content will be found with a probability tending to 0. Thus, since by construction we have $\sum_m P_{\text{in}}(m) = C$, only C contents can have maximum interrequest time smaller than $T_{C,\text{sup}}$. This concludes the proof.

C. EXACT CALCULATION OF $P_{IN(M)}$ FOR 2-LRU

For simplicity in this appendix, whenever not strictly necessary, we omit the dependency of variables on m. We define as cycle the time interval between two visits at state $(1, 1)$ (i.e., the time interval between two requests for object m that bring the system to state $(1, 1)$). Observe that, by construction, the cycles are i.i.d. We consider a generic cycle starting at time $t = 0$ (thus, by construction, a request for m arrives at time $t = 0$). Let R_1 be the time of the first request for object m after $t = 0$. We have the following possibilities:

—$R_1 \leq T_C^1$: At time R_1^-, the system is still in state $(1, 1)$, and consequently $\mathbb{E}[T_{\text{cycle}} \mid R_1 < T_C^1] = \mathbb{E}[R_1 \mid R_1 < T_C^1]$.

—$T_C^1 < R_1 \leq T_C^2$: In this case, at time $t = T_C^1$, the system enters state $(0, 1)$, where it is found at R_1^-; thus, the request at R_1 brings the system again in state $(1, 1)$. In this case, $\mathbb{E}[T_{\text{cycle}} \mid T_C^1 < R_1 \leq T_C^2] = \mathbb{E}[R_1 \mid T_C^1 < R_1 \leq T_C^2]$.

—$R_1 > T_C^2$: In this last case, the analysis is more complicated. At time T_C^1, the system goes to state $(0, 1)$, and at time T_C^2, it enters state $(0, 0)$. At time $t = R_1$, for effect of

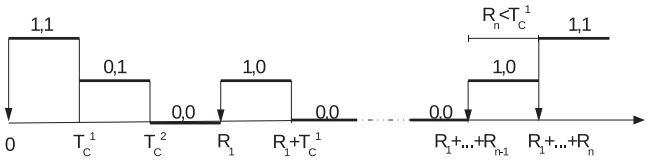

Fig. 15. Illustration of the cycle analysis used for deriving Equation (20). Vertical arrows represent incoming requests for the content.

the arrival of the first request, the system enters $(1, 0)$. Now, if the following request arrives before $R_1 + T_C^1$, the system goes back to state $(1, 1)$ and the cycle terminates. Otherwise, the system at time $R_1 + T_C^1$ enters state $(0, 0)$ again and the following request brings it to state $(1, 0)$ again. The cycle ends upon the arrival of the first request for object m that follows the previous one by less than T_C^1. Figure 15 better illustrates this situation.

Thus, if we denote by R_i the ith interrequest time, and with $n \sim \text{Geom}(p_1)$, $p_1 = \mathbb{P}(R \le T_C^1) = 1 - e^{-\lambda_m T_C^1}$ and $p_2 = \mathbb{P}(R \le T_C^2) = 1 - e^{-\lambda_m T_C^2}$, we can write the following in this case:

$$\begin{aligned}
\mathbb{E}[T_{\text{cycle}} \mid R_1 > T_C^2] &= \mathbb{E}\big[R_1 \mid R_1 > T_C^2\big] + \mathbb{E}\big[R_n \mid R_n \le T_C^1\big] + \mathbb{E}\left[\sum_{i=0}^{n-1} R_i \mid R_i > T_C^1\right] \\
&= \mathbb{E}\big[R_1 \mid R_1 > T_C^2\big] + \mathbb{E}\big[R_n \mid R_n \le T_C^1\big] + \mathbb{E}[n]\mathbb{E}\big[R_i \mid R_i > T_C^1\big] \\
&= \mathbb{E}\big[R_1 \mid R_1 > T_C^2\big] + \frac{\mathbb{E}\big[R_n, R_n \le T_C^1\big]}{\mathbb{P}\big(R_n \le T_C^1\big)} + \frac{1-p_1}{p_1}\frac{\mathbb{E}\big[R_i, R_i > T_C^1\big]}{\mathbb{P}\big(R_i > T_C^1\big)} \\
&= \mathbb{E}\big[R_1 \mid R_1 > T_C^2\big] + \frac{\mathbb{E}\big[R_n, R_n \le T_C^1\big]}{p_1} + \frac{1-p_1}{p_1}\frac{\mathbb{E}\big[R_i, R_i > T_C^1\big]}{1-p_1} \\
&= \mathbb{E}\big[R \mid R > T_C^2\big] + \frac{\mathbb{E}[R]}{p_1}.
\end{aligned}$$

Considering the other cases as well, we have that

$$\begin{aligned}
\mathbb{E}[T_{\text{cycle}}] &= \mathbb{E}\big[R_1 \mid R_1 < T_C^2\big]\mathbb{P}\big(R_1 \le T_C^2\big) + \left(\mathbb{E}\big[R_1 \mid R_1 > T_C^2\big] + \frac{\mathbb{E}[R]}{p_1}\right)\mathbb{P}\big(R_1 > T_C^2\big) \\
&= \mathbb{E}[R] + \frac{\mathbb{E}[R]}{p_1}(1 - p_2). \qquad (20)
\end{aligned}$$

Turning our attention to $\mathbb{E}[T_{\text{ON}}]$, which is the average time within a cycle during which content m is stored in the second (physical) cache, we have that

$$\mathbb{E}[T_{\text{ON}}] = \mathbb{E}\big[\min\big(R_1, T_C^2\big)\big] = \mathbb{E}\big[R_1 \mid R_1 < T_C^2\big]\mathbb{P}\big(R_1 < T_C^2\big) + T_C^2\,\mathbb{P}\big(R_1 \ge T_C^2\big). \qquad (21)$$

Last, we can obtain $p_{\text{in}}(m)$ as

$$p_{\text{in}}(m) = \frac{\mathbb{E}[T_{\text{ON}}(m)]}{\mathbb{E}[T_{\text{cycle}}(m)]}.$$

D. PROOF OF K-LRU → LFU

For simplicity, we limit ourselves to the IRM traffic model. An analogous result can be derived under *renewal* traffic along the same lines. First, we recall that sequence $\{T_C^i\}_{i=1}^k$ is increasing. We prove that $T_C^* = \sup_{k\to\infty} T_C^k = +\infty$. Assume, by contradiction,

that T_C^* is finite. Now, a necessary condition for content m to be in the cache at time t is that a request arrived at $\tau_1 \in (t - T_C^k, t]$. This request, in turn, must have necessarily generated a hit either in cache k or in cache $k-1$. Consequently, a previous request must have arrived at $\tau_2 \in (\tau_1 - T_C^k, \tau_1]$. Iterating back, we generate a chain of k requests for object m requests with interrequest time smaller than T_C^k, which is necessary for object m to be found in cache k at time t. The probability of observing such a chain is bounded by $(1 - e^{-\lambda_m T_C^*})^k$; this probability goes to zero when $k \to \infty$, independently on λ_m, leading to a contradiction. Recall that, by construction, $\sum p_{\text{in}}(m,k) = C$. Thus, we can conclude that $\lim_{k\to\infty} T_C = +\infty$. Recalling the expression in Equation (11),

$$p_{\text{in}}(m,i) = (1 - e^{-\lambda_m T_C^i})[p_{\text{in}}(m,i) + (p_{\text{in}}(m,i-1))(1 - p_{\text{in}}(m,i))],$$

we can easily prove that (i) $p_{\text{in}}(m,i)$ is increasing with respect to λ_m for any i (by induction, over i); and (ii) Equation (11), for sufficiently large T_C^i, is a contraction mapping over $[\epsilon, 1]$ for any $\epsilon > 0$.

Thus, $\lim_{k\to\infty} p_{\text{in}}(m,k)$ exists and it is necessarily the fixed point $p_{\text{in}}^*(m)$ of Equation (11). The assertion immediately follows, since $p_{\text{in}}^*(m) \in \{0,1\}$.

The extension to the non-IRM case, under the assumption that the support of the interrequest time distribution is unbounded, and that for any m_1 and m_2, with $\lambda_{m_1} < \lambda_{m_2}$, $\lim_{t\to\infty} \frac{1-F(m_1,t)}{1-F(m_2,t)} > 1$, follows the same lines.

REFERENCES

V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. 1996. Characterizing reference locality in the WWW. In *IEEE PDIS*.

D. S. Berger, P. Gland, S. Singla, and F. Ciucu. 2014. Exact analysis of TTL cache networks. *Performance Evaluation* 79, 2–24.

Giuseppe Bianchi, Andrea Detti, Alberto Caponi, and Nicola Blefari Melazzi. 2013. Check before storing: What is the performance price of content integrity verification in LRU caching? *SIGCOMM Computer Communication Review* 43, 3, 59–67. DOI:http://dx.doi.org/10.1145/2500098.2500106

L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. 1999. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM*. 126–134.

Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, and Diego Perino. 2011. Modeling data transfer in content-centric networking *(ITC)*. 8. http://dl.acm.org/citation.cfm?id=2043468.2043487.

Meeyoung Cha, Haewoon Kwak, P. Rodriguez, Yong-Yeol Ahn, and Sue Moon. 2009. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networking* 17, 5, 1357–1370. DOI:http://dx.doi.org/10.1109/TNET.2008.2011358

Hao Che, Ye Tung, and Z. Wang. 2002. Hierarchical web caching systems: Modeling, design and experimental results. *IEEE Journal on Selected Areas in Communications* 20, 7, 1305–1314.

E. Coffman and P. Denning. 1973. *Operating Systems Theory*. Prentice-Hall, Englewood Cliffs, NJ.

A. Dan and D. Towsley. 1990. An approximate analysis of the LRU and FIFO buffer replacement schemes. *SIGMETRICS Performance Evaluation Review* 18, 1, 143–152. DOI:http://dx.doi.org/10.1145/98460.98525

Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M. Munafò, and Dario Rossi. 2011. Experiences of Internet traffic monitoring with Tstat. *IEEE Network* (2011).

Nicaise Choungmo Fofack. 2014. *On Models for Performance Analysis of a Core Cache Network and Power Save of a Wireless Access Network*. Thesis. Université Nice Sophia Antipolis. Retrieved March 25, 2016 from https://tel.archives-ouvertes.fr/tel-00968894.

N. Choungmo Fofack, M. Dehghan, D. Towsley, M. Badov, and D. L. Goeckel. 2014a. On the performance of general cache networks. In *VALUETOOLS'14*. 106–113. DOI:http://dx.doi.org/10.4108/icst.Valuetools.2014.258168

Nicaise Choungmo Fofack, Philippe Nain, Giovanni Neglia, and Don Towsley. 2014b. Performance evaluation of hierarchical TTL-based cache networks. *Computer Networks* 65, 212–231. DOI:http://dx.doi.org/10.1016/j.comnet.2014.03.006

R. Fonseca, V. Almeida, M. Crovella, and B. Abrahao. 2003. On the intrinsic locality of web reference streams. In *INFOCOM*.

Christine Fricker, Philippe Robert, and James Roberts. 2012a. A versatile and accurate approximation for LRU cache performance *(ITC)*.

Christine Fricker, Philippe Robert, James Roberts, and Nada Sbihi. 2012b. Impact of traffic mix on caching performance in a content-centric network. In *IEEE NOMEN Workshop*.

Massimo Gallo, Bruno Kauffmann, Luca Muscariello, Alain Simonian, and Christian Tanguy. 2012. Performance evaluation of the random replacement policy for networks of caches. *SIGMETRICS Performance Evaluation Reviews* 40, 1, 395–396. DOI:http://dx.doi.org/10.1145/2318857.2254810

M. Garetto, E. Leonardi, and S. Traverso. 2015. Efficient analysis of caching strategies under dynamic content popularity. In *Infocom'15*.

Erol Gelenbe. 1973. A unified approach to the evaluation of a class of replacement algorithms. *IEEE Transactions on Computers* 22, 6, 611–618.

Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. 2009. Networking named content. In *ACM CoNEXT*.

P. Jelenković and A. Radovanović. 2003. Asymptotic insensitivity of least-recently-used caching to statistical dependency. In *INFOCOM*.

Predrag R. Jelenković. 1999. Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. *Annals of Applied Probability* 9, 2, 430–464.

Predrag R. Jelenković and Xiaozhu Kang. 2008. Characterizing the miss sequence of the LRU cache. *SIGMETRICS Performance Evaluation Review* 36, 2, 119–121. DOI:http://dx.doi.org/10.1145/1453175.1453203

Predrag R. Jelenković and Ana Radovanović. 2004. Least-recently-used caching with dependent requests. *Theoretical Computer Science* 326, 13, 293–327.

Predrag R. Jelenković and Ana Radovanović. 2008. The persistent-access-caching algorithm. *Random Structures and Algorithms* 33, 2, 219–251.

Predrag R. Jelenković and Mark S. Squillante. 2006. Critical sizing of LRU caches with dependent requests. *Journal of Applied Probability* 43, 4, 1013–1027.

Wenjie Jiang, Stratis Ioannidis, Laurent Massoulié, and Fabio Picconi. 2012. Orchestrating massively distributed CDNs. In *ACM CoNEXT*.

Shudong Jin and A. Bestavros. 2000. Sources and characteristics of web temporal locality. In *IEEE MASCOTS*.

Theodore Johnson and Dennis Shasha. 1994. 2Q: A low overhead high performance buffer management replacement algorithm. In *VLDB*.

W. F. King. 1971. *Analysis of Paging Algorithms*. Retrieved March 25, 2016 from http://books.google.it/books?id=KTvaPgAACAAJ.

Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. 2006. The LCD interconnection of LRU caches and its analysis. *Performance Evaluation* 63, 7, 609–634. DOI:http://dx.doi.org/10.1016/j.peva.2005.05.003

D. Lee, J. Choi, J. H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim. 2001. LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE Transactions on Computers* 50, 12, 1352–1361. DOI:http://dx.doi.org/10.1109/TC.2001.970573

Zhe Li, G. Simon, and A. Gravey. 2012. Caching policies for in-network caching. In *ICCCN*.

Felipe Olmos, Bruno Kauffmann, Alain Simonian, and Yannick Carlinet. 2014. Catalog dynamics: Impact of content publishing and perishing on the performance of a LRU cache. In *ITC*. IEEE, 1–9.

Elizabeth J. O'Neil, Patrick E. O'Neil, and Gerhard Weikum. 1993. The LRU-K page replacement algorithm for database disk buffering. In *SIGMOD'93*. ACM, New York, NY, 297–306. DOI:http://dx.doi.org/10.1145/170035.170081

Ioannis Psaras, Richard G. Clegg, Raul Landa, WeiKoong Chai, and George Pavlou. 2011. Modelling and evaluation of CCN-caching trees. In *IFIP NETWORKING*. Lecture Notes in Computer Science, Vol. 6640. Springer, Berlin, 78–91.

E. J. Rosensweig, J. Kurose, and D. Towsley. 2010. Approximate models for general cache networks. In *INFOCOM*.

E. J. Rosensweig, D. J. Menache, and J. Kurose. 2013. On the steady-state of cache networks. In *INFOCOM*.

Giuseppe Rossini and Dario Rossi. 2014. Coupling caching and forwarding: Benefits, analysis, and implementation. In *ICN'14*. ACM, New York, NY, 127–136. DOI:http://dx.doi.org/10.1145/2660129.2660153

Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. 2013. Temporal locality in today's content caching: Why it matters and how to model it. *SIGCOMM Computer Communication Review* 43, 5, 5–12. DOI:http://dx.doi.org/10.1145/2541468.2541470

Received November 2014; revised September 2015; accepted February 2016