POLITECNICO DI TORINO
Repository ISTITUZIONALE

A unified approach to the performance analysis of caching systems

Original
A unified approach to the performance analysis of caching systems / Martina, V.; Garetto, M.; Leonardi, Emilio. - ELETTRONICO. - (2014), pp. 2040-2048. (Infocom 2014 Toronto, (CN) April 2014) [10.1109/INFOCOM.2014.6848145].

Availability:
This version is available at: 11583/2551749 since:

Publisher:
IEEE - INST ELECTRICAL ELECTRONICS ENGINEERS INC

Published
DOI:10.1109/INFOCOM.2014.6848145

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

07 May 2026

A unified approach to the performance analysis of caching systems

Valentina Martina *, Michele Garetto †, Emilio Leonardi *
* Dipartimento di Elettronica, Politecnico di Torino, Torino, Italy
† Dipartimento di Informatica, Università di Torino, Torino, Italy

***Abstract*—We propose a unified methodology to analyse 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, 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.**

I. Introduction and paper contributions

In the past few years the performance of caching systems, one of the most traditional and widely investigated topic 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 in the Internet. Thanks to an 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 [1]. By caching contents close to the users, we jointly reduce network traffic and improve user-perceived experience.

The fundamental role played by caching systems in the Internet goes beyond existing content delivery networks, as consequence of the gradual shift from the traditional host-to-host communication model to the new host-to-content paradigm. Indeed, a novel Information Centric Network (ICN) architecture has been proposed for the future Internet to better respond to the today and future (according to predictions) traffic characteristics [2]. In this architecture, caching becomes an 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. Unfortunately, evaluating the performance of cache networks is hard, considering that the computational cost to exactly analyse just a single LRU (Least Recently Used) cache, grows exponentially with both the cache size and the number of contents [3], [4]. Nevertheless, several approximations have been proposed over the years [4], [5], [6], [7], [8], [9] which can accurately predict the cache performance at an affordable computational cost.

The main drawback of existing analytical techniques is their rather limited scope. Indeed, many of them target only specific caching policies (mainly LRU and FIFO) under simplifying traffic conditions (most of previous work relies on the Independent Reference Model, [10]), while the analysis of cache networks has only recently been attempted (essentially for LRU) [11], [7], [8], [12]. We refer the reader to [13] for an extended survey of related work.

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 approximation) has much broader applicability than the particular context in which it was originally proposed (*i.e.*, a single LRU cache under IRM traffic), and can actually provide the key to develop a general methodology to analyse a variety of caching systems.

In particular, in this paper 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 analysed so far (including a novel multi-stage scheme called k-LRU), implementing different insertion/eviction policies; ii) more general traffic conditions than the traditional IRM, so as to capture the effects of temporal locality in the requests arrival process; iii) a scalable and accurate technique to analyse interconnected caches that goes beyond the standard Poisson assumption adopted so far, allowing us to consider large-scale networks implementing also coordinated replication strategies (such as leave-copy-down).

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

Furthermore, under the small cache regime (*i.e.*, cache size small with respect to the content catalog size), which is of special interest for ICN, our expressions can be further simplified, leading to simple closed-form expressions of the cache hit probability, and 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 of a general performance evaluation tool for caching systems.

M. Garetto was supported by the AMALFI Project (Università di Torino/Compagnia di San Paolo). Copyright 978-1-4799-3360-0/14/$31.00 ©2014 IEEE

II. System assumptions

A. 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 [10]. The IRM is based on the following fundamental assumptions: i) users request items from a fixed catalogue of M object; 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 definition, the IRM completely ignores all temporal correlations in the sequence of requests. In particular, it does not take into account a key feature of real traffic usually referred to as *temporal locality*, *i.e.*, the fact that, if an object is requested at a given point in time, then it is more likely that the same object will be requested again in the near future.

This characteristic of real sequences of requests, and its beneficial effect on cache performance, are well known, especially in the context of computer memory architecture and web traffic [14]. Indeed, several extensions of IRM have been proposed to incorporate temporal locality in the traffic model. Existing approaches [10], [15], [9] typically assume that the request process for each object is stationary (i.e, either a renewal process or a Markov- or semi-Markov-modulated Poisson process).

To account for temporal locality, in this paper we will consider, whenever possible, 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 inter-request time distribution. Let $F_R(m,t)$ be the cdf of the inter-request 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 at the cache. Note that, by adopting an object popularity law analogous to the one considered by the IRM, we also have $\lambda_m = \Lambda p_m$.

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

B. Popularity law

Traffic models like the IRM (and its generalizations) are commonly used in combination with a Zipf-like law of object popularity, which is frequently observed in traffic measurements and widely adopted in performance evaluation studies [16].

In its simplest form, Zipf's law states that the probability to request the i-th most popular item is proportional to $1/i^\alpha$, where the exponent α depends on the considered system (especially on the type of objects) [17], and plays a crucial role on the resulting cache performance. Estimates of α reported in the literature for various kinds of systems range between .65 and 1.

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

C. 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 policy statically stores in the cache the C most popular contents (assuming 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* item is evicted, *i.e.*, the object which 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 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. Notice that this scheme is different from LRU because requests finding the object in the cache do not 'refresh' the timer 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 (storing actual objects), indexed by k, requests have to traverse a chain of $k-1$ virtual caches put in front of it, which store only object hashes and perform meta-cache operations on them. Upon arrival of a request, the request enters cache i if its hash is already stored in cache $i-1$ (or if $i = 1$). The eviction policy at all caches is LRU. For simplicity, we will restrict ourselves to the case in which all caches have the same size (expressed either in terms of objects or hashes).
- **k-RANDOM**: it works exactly like k-LRU, with the only difference that the eviction policy at each cache is RANDOM.

We remark that LRU has been widely adopted, since it provides good performance while being reasonably simple to implement. RANDOM and FIFO have been considered as viable alternative to LRU in the context of ICN, as their hardware implementation in high-speed routers is even simpler. The q-LRU policy and multi-stage 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 parameter[1] (the number of caches k), providing significant improvements

[1]More sophisticated insertion policies such as the persistent-access-caching algorithm [18] obtain a filtering effect similar to k-LRU but require more parameters which are not easy to set.

over LRU even for very small k (much of the possible gain is already achieved by $k = 2$).

D. 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 toward 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:

- **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 (if any).

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

III. The Che approximation

We briefly recall Che's approximation for LRU under the classical IRM [5]. Consider a cache capable of storing C distinct 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, *i.e.*, the time since the last request after which object m will be evicted from the cache (if the object is not again requested 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 in [19], where 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 negligible when the catalogue size is sufficiently large.

The reason why Che's approximation greatly simplifies the analysis of caching systems is because it allows to decouple 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.

More in detail, thanks to Che's approximation, we can state 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, *i.e.*, if at least a 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 given by:

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

As immediate consequence of PASTA property for Poisson arrivals, observe that $p_{\text{in}}(m)$ represents, by construction, also the hit probability $p_{\text{hit}}(m)$, *i.e.*, 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:

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

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

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

IV. 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 be used to predict cache performance also under *renewal* traffic, and to analyse policies other than LRU.

A. LRU *under* renewal *traffic*

The extension of Che's approximation to the *renewal* traffic model is conceptually simple although it requires some care. Indeed, observe that, under a general request process, we can not 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 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 $\hat{F}_R(T_C)$ is the pdf of the *age* associated to object-m inter-request 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 inter-request time does not exceed T_C, yielding:

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

B. q-LRU *under IRM and* renewal *traffic*

We now analyse 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:

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

Solving the above expression for $p_{\text{in}}(m)$, we get:

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

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:

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

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

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

Regarding the q-LRU policy, Che's approximation allows to establish the following interesting property as $q \to 0$, whose proof is reported in [13].

Theorem 1: The q-LRU policy tends asymptotically to LFU as the insertion probability goes to zero.

C. 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. Indeed 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 Erlang-B formula (exploiting PASTA):

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

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

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

Proposition 1: Under IRM traffic, the performance of RANDOM and FIFO (in terms of hit probability) are the same.

This result was originally obtained by Gelenbe [20] using a totally different approach.

Note that, under FIFO policy, we can assume T_C to be a constant, in perfect analogy to LRU. Indeed, T_C is still equal to the time needed to observe the requests for C different 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. Indeed, under RANDOM an object is evicted with probability $1/C$ upon arrival of each request for an object which 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 analysed, under *renewal traffic*, employing basic queuing theory. Probability p_{hit} can be easily obtained as the loss probability of the G/M/1/0 queue. Solving the Markov chain representing the number of customers in the system at arrival times, we obtain:

$$p_{\text{hit}}(m) = M_R(m, -1/\mathbb{E}[T_C])$$

where $M_R(m, \cdot)$ is the moment generating function of object-m inter-request 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 such a 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 inter-request 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, we have by standard renewal theory $p_{\text{in}}(m) = \mathbb{E}[T_{\text{ON}}]/\mathbb{E}[T_{\text{cycle}}]$. Now, we know that $\mathbb{E}[T_{\text{cycle}}] = 1/\lambda_m$. For $\mathbb{E}[T_{\text{ON}}]$, we obtain:

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

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

D. 2-LRU

We now move to the novel 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 analyse 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:

$$p_{\text{hit}}(m) = p_{\text{in}}(m) \approx$$
$$\approx (1 - e^{-\lambda_m T_C^2})[p_{\text{hit}}(m) + (1 - e^{-\lambda_m T_C^1})(1 - p_{\text{hit}}(m))] \quad (8)$$

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). Indeed, in this case the states of cache 2 and meta-cache 1 tends to de-synchronize, since an 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 $T_C^1 \ge T_C^2$).

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 Fig. 1, where each state is denoted by a pair of binary variables indicating the presence of object m in meta-cache 1 and cache

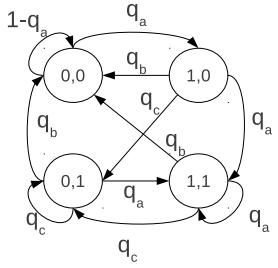

Fig. 1. DTMC describing the dynamics of an object in 2-LRU, sampled at request arrival times.

2, respectively. Solving the DTMC, we get:

$$p_{\text{hit}}(m) = p_{\text{in}}(m) = 1 - \frac{(1+q_a)q_b}{q_a+q_b} \tag{9}$$

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:

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

Also the refined model can be generalized to *renewal* traffic, observing that object-m dynamics in the system, sampled at request arrivals, are still described by a Markov Chain with exactly the same structure as in Fig. 1 (only the expressions of transition probabilities change in an obvious way). Thus we obtain:

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

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 our technical report [13].

E. 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 (8) 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:

$$\begin{aligned} &p_{\text{hit}}(i, m) = p_{\text{in}}(i, m) \approx \\ &(1 - e^{-\lambda_m T_C^i})[p_{\text{hit}}(i, m) + (p_{\text{hit}}(i-1, m))(1 - p_{\text{hit}}(i, m))] \end{aligned} \tag{10}$$

The generalization to *renewal* traffic is straightforward.

At last, for large k we can state:

Theorem 2: According to (10) 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. The proof is reported in [13].

F. k-RANDOM

Also k-RANDOM can be analysed under Che's approximation assuming exponential sojourn times in the caches. In particular, 2-RANDOM can be exactly described (under Che's approximation) by a simple four-states continuous time Markov chain representing the dynamics of object m in the system. Due to space constrains we omit the details.

G. 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. This is especially useful to understand the dependency of p_{in} and p_{hit} on the object arrival rate λ_m (and thus its popularity), obtaining interesting insights into the performance of the various caching policies. Due to space limitations, we restrict ourselves to IRM traffic, however we emphasize that a similar approach can be generalized to *renewal* traffic. We obtain:

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

Previous expressions permit us immediately to rank the performance of the considered policies in the small cache regime. Indeed, 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 $n \geq 2$, 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 (which makes the dependency of $p_{\text{hit}}(m)$ on λ_m slightly super-linear). At last LRU slightly outperforms RANDOM/FIFO because its negative quadratic term has a smaller coefficient.

H. 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.

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

Fig. 2 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 (10) 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 which do not filter any request. The impact of the eviction policy, instead, appears

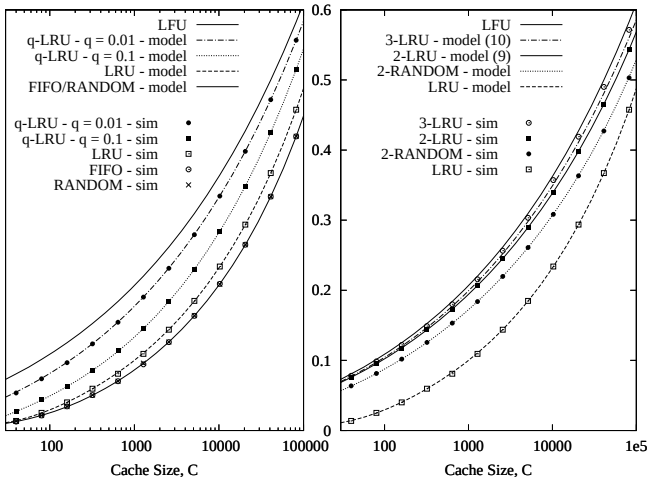

Fig. 2. Hit probability vs cache size, for various caching policies, under IRM.

to be much weaker, with LRU providing moderately better performance than RANDOM/FIFO.

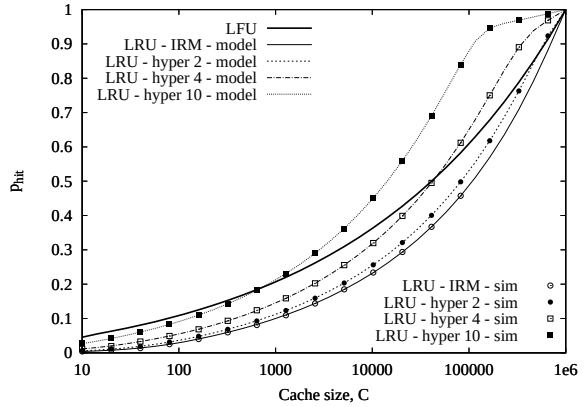

Fig. 3. Hit probability vs cache size, for LRU, under different degrees of temporal locality.

Fig. 3 shows the impact of temporal locality on caching performance: LRU performance is evaluated under *renewal* traffic in which object inter-arrival times are distributed according to a second order hyper-exponential with branches $\lambda_m^1 = z\lambda_m$ and $\lambda_m^2 = \lambda_m/z$ (hereinafter, we will call hyper-z such distribution), so that increasing values of z results into 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.

Fig. 3 also shows that LFU is no longer optimal when traffic does not satisfy the IRM. This 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 into suboptimal performance.

Fig. 4 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).

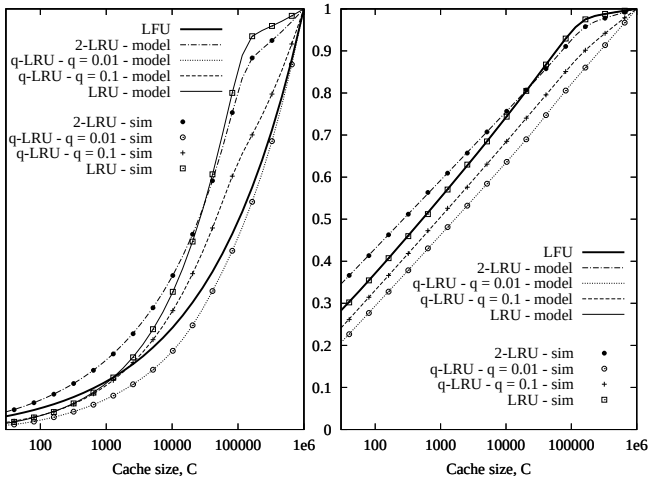

Fig. 4. Hit probability vs cache size, for various caching policies, under hyper-10 traffic, in the case of $\alpha = 0.7$ (left plot) or $\alpha = 1$ (right plot).

We 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 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 provides very good performance also in the presence of strong temporal locality. This 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.

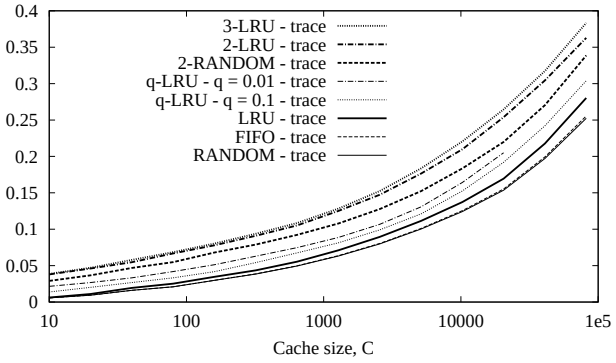

Fig. 5. Hit probability vs cache size, for various caching policies, under real trace of Youtube video requests.

To further validate the design insights gained by our analysis, we have also 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 analysing TCP flows by means of Tstat, an open-source traffic monitoring tool developed at Politecnico di Torino [21]. During a period of 35 days in year 2012, from March 20th to April 25th, we recorded in total 3.8M of requests, for 1.76M of videos, coming from 31124 distinct IP addresses.

Fig. 5 reports the hit probability achieved by different caching

schemes[2]. 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 (q-LRU), while additional meta-caches provide only minor improvements; iii) the impact of the eviction policy is not significant, especially when caches are small with respect to the catalogue size.

V. Cache networks

In a typical cache network, caches forward their miss stream (*i.e.*, requests which have not found the target object) to other caches. Let us briefly recall the standard approach that has been proposed in the literature to analyse 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 later equation (12).

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 multi-variable fixed-point approach is then used to solve the entire system (see [7] 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 Sec. II-D. To describe our improved technique, it is sufficient to consider the simple case of just two caches (tandem network). Indeed, the extension of our method to general network is straightforward.

Moreover, we will limit ourselves to the case in which the traffic produced by the users satisfies the IRM model (*i.e.*, the exogenous process of requests for each object is indeed 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.

We remark that an alternative approach to ours has been recently proposed in [12] for cache networks with feed-forward topology, implementing TTL-based eviction policies. Their approach, which can be used to analyse the performance LRU, RANDOM and FIFO under the Che approximation, essentially consists in characterizing the inter-request process arriving at non-ingress caches through a two steps procedure: i) the miss stream of (ingress) caches is exactly characterized as a renewal process with given distribution; ii) by exploiting known results on the superposition of independent renewal processes, the inter-request time distribution at non-ingress caches is obtained. Although the approach in [12] is very elegant, and it 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; ii) it can be hardly generalized to general mesh networks (non feed-forward). Our approach is somehow complementary to the one proposed in [12] since, while it applies only to IRM traffic, it is definitely more scalable and readily applicable to networks with general topology.

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

A. 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

$$p_{\text{hit}}(1,m) = p_{\text{in}}(1,m) = 1 - e^{-\lambda_m(1)T_C^1}. \tag{11}$$

The arrival process of request for object m at the second cache is not Poisson. It is, instead, an ON-OFF modulated Poisson process, where 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 $\overline{\lambda}_m(2) = \lambda_m(1)(1-p_{\text{in}}(1,m))$ 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:

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

Essentially, the improvement with respect to the standard approximation consists in the term $T_C^2 - T_C^1$ in the above equation, in place of T_C^2. This simple modification, however, can make a huge difference in the resulting hit probability at cache 2 (we omit example results due to lack of space). If, instead, $T_C^2 < T_C^1$, we clearly have $p_{\text{hit}}(2,m) = 0$.

Note that the above 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.

B. 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 (5) to analyze its behavior.

To evaluate $p_{\text{hit}}(2,m)$, we again 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 $T_C^2 > T_C^1$ 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; ii) it arrived at cache 2 in the time interval $[t-T_C^1, t]$, without producing an insertion in cache 1, or it arrived at cache 2 in the interval $[t-T_C^2, t-T_C^1]$. Thus, we obtain:

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

Note that, in the previous expression, we employ the full rate $\lambda_m(1)$ for requests arriving at cache 1 in the interval $[t-T_C(1), t]$, since by construction cache 1 does not contain m at t^-. On the other hand, we approximate the arrival process of requests for m at cache 2 with a Poisson process of rate $\overline{\lambda}_m(2)$ over the interval $[t-T_C^2, t-T_C^1]$.

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

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

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

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

C. Leave-copy-down

This strategy is more complex to analyse, since now the dynamics of cache 1 and cache 2 depend mutually on each other. Indeed, 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:

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

Observe in the previous expression that we have assumed the states of cache 1 and cache 2 to be independent; on the other hand, similarly to what we have done before, we write:

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

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:

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

Indeed, since at time t^- cache 1 does not store the object by construction, either the previous request arrived at cache 2 in $[t-T_C^2, t-T_C^1]$, or it arrived in $[t-T_C^1, t]$ but did not trigger an insertion in cache 1 because object m was not found in cache 2. Similarly, if $T_C^2 < T_C^1$:

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

At last we remark that cache networks implementing LCD have been previously considered in [22] for the special case of tandem topologies. We emphasize that the method proposed in this paper provides a significantly simpler and higher scalable alternative to the approach devised in [22], by capturing in a simple yet effective way existing correlations between caches' states, while reducing the number of interdependent parameters that must be estimated through the iterative procedure.

D. Extension to general cache networks

Our approach, which has been described above 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 leave-copy-everywhere scheme. Let $r_{j,i}$ be the fraction of requests for object m which are forwarded from cache j to cache i (in the case of a miss in cache j).

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

$$\overline{\lambda}_m(i) = \sum_j \overline{\lambda}_m(j)(1-p_{hit}(j,m))r_{j,i} \tag{12}$$

and we can immediately express:

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

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:

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

where $A_{i,j} = r_{j,i}\overline{\lambda}_m(j)(1-p_{in}(j,m))\max(0, T_C^i - T_C^j) + \sum_{k\neq j} r_{k,i}\overline{\lambda}_m(k)(1-p_{in}(k,m))T_C^i$. The expression for $p_{\text{hit}}(i,m)$ can then be obtained de-conditioning with respect to j.

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

E. 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.

Fig 6 compares the performance of the different replication strategies that we have analysed, 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

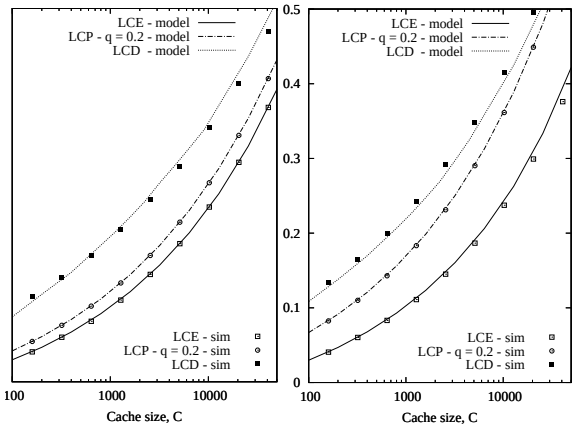

Fig. 6. Hit probability vs cache size, for various replication strategies, in the case of a chain of 6 caches, under IRM traffic. Hit probability of the first cache (left plot) and total hit probability of the network (right plot).

degree of correlation among caches (and 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 the dual of k-LRU for cache networks).

At last, we consider a very large topology comprising 1365 caches, corresponding to a 4-ary regular tree with 6 levels. Such 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. Fig. 7 reports the total hit probability achieved in this large network, for two traffic scenarios (analytical results only).

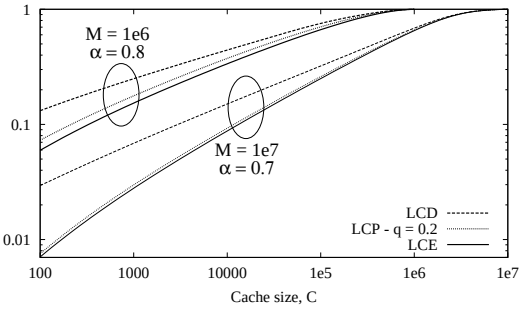

Fig. 7. Hit probability vs cache size, for various replication strategies, in the case of a tree topology with 1365 caches, for two traffic scenarios

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

VI. Conclusions

The main goal of this paper 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 analysed 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 which 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.

References

[1] W. Jiang, S. Ioannidis, L. Massoulié, and F. Picconi, "Orchestrating massively distributed CDNs," in *ACM CoNEXT*, 2012.
[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT 2009*.
[3] K. W. King, "Analysis of paging algorithms," in *IFIP Congress*, 1971.
[4] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," *SIGMETRICS Perform. Eval. Rev.*, vol. 18, pp. 143–152, Apr. 1990.
[5] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: modeling, design and experimental results," *IEEE JSAC*, vol. 20, no. 7, pp. 1305–1314, 2002.
[6] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for lru cache performance," ITC 2012.
[7] E. Rosensweig, J. Kurose, and D. Towsley, "Approximate Models for General Cache Networks," in *INFOCOM, 2010*.
[8] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, "Performance evaluation of the random replacement policy for networks of caches," *SIGMETRICS Perf. Eval. Rev.*, vol. 40(1), pp. 395–396, 2012.
[9] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi, "Check before storing: what is the performance price of content integrity verification in lru caching?," *Comput. Comm. Rev.*, vol. 43, pp. 59–67, July 2013.
[10] E. Coffman and P. Denning., *Operating Systems Theory.* Englewood Cliffs (NJ): Prentice-Hall, 1973.
[11] E. Rosensweig, D. J. Menache, and J. Kurose, "On the Steady-State of Cache Networks," in *INFOCOM*, 2013.
[12] N. Fofack, P. Nain, G. Neglia, and D. Towsley, "Analysis of ttl-based cache networks," in *VALUETOOLS*, 2012.
[13] Companion technical report, available at http://arxiv.org/abs/1307.6702.
[14] R. Fonseca, V. Almeida, M. Crovella, and B. Abrahao, "On the intrinsic locality of web reference streams," in *IEEE INFOCOM*, 2003.
[15] K. Kylkoski and J. Virtamo, "Cache replacement algorithms for the renewal arrival model," in *Fourteenth Nordic Teletraffic Seminar, NTS-14*, (Copenhagen, Denmark), pp. 139–148, Aug. 1998.
[16] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems," *IEEE/ACM Trans. on Netw.*, vol. 17(5), pp. 1357–1370, 2009.
[17] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *NOMEN*, 2012.
[18] P. R. Jelenković and A. Radovanović, "The persistent-access-caching algorithm," *Random Struct. Algorithms*, vol. 33, pp. 219–251, Sept. 2008.
[19] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *ITC*, 2012.
[20] E. Gelenbe, "A unified approach to the evaluation of a class of replacement algorithms," *IEEE Trans. Comput.*, vol. 22, pp. 611–618, June 1973.
[21] A. Finamore, M. Mellia, M. Meo, M. M. Munafò, and D. Rossi, "Experiences of Internet traffic monitoring with Tstat," *IEEE Network*, 2011.
[22] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Perform. Eval.*, vol. 63, July 2006.