Check Before Storing: What is the Performance Price of Content Integrity Verification in LRU Caching?

Giuseppe Bianchi, Andrea Detti, Alberto Caponi, Nicola Blefari-Melazzi
CNIT / Univ. Roma Tor Vergata
name.surname@uniroma2.it

ABSTRACT

In some network and application scenarios, it is useful to cache content in network nodes on the fly, at line rate. Resilience of in-network caches can be improved by guaranteeing that all content therein stored is valid. Digital signatures could be indeed used to verify content integrity and provenance. However, their operation may be much slower than the line rate, thus limiting caching of cryptographically verified objects to a small subset of the forwarded ones. How this affects caching performance? To answer such a question, we devise a simple analytical approach which permits to assess performance of an LRU caching strategy storing a randomly sampled subset of requests. A key feature of our model is the ability to handle traffic beyond the traditional Independent Reference Model, thus permitting us to understand how performance vary in different temporal locality conditions. Results, also verified on real world traces, show that content integrity verification does not necessarily bring about a performance penalty; rather, in some specific (but practical) conditions, performance may even improve.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Network Architecture and Design; C.4 [**Performance of Systems**]: Modeling techniques

General Terms

Performance, Security, Theory

Keywords

Caching, Information Centric Networks, Digital signatures, Performance Modeling

1. INTRODUCTION

In-network caching is nowadays central in the "Internet of Content", founded on emerging Information Centric Networking (ICN) frameworks [2]. The distinguishing feature of ICN is the ability to deliver a named block of data (a whole object or a *chunk* within a larger content item), irrespective of where (i.e., on which specific server) such data block is specifically stored. In ICNs, content-oriented data chunks[1] are atomic units explicitly addressed *by-name* [23, 21], and caching is carried out *systematically* and *on-the-fly*, without the need to deploy cumbersome processing tasks such as HTTP header parsing. Moreover, ICN caching has a *universal* nature[20]: not only a subset of paying content providers (as in Content Distribution Networks [28]), but all Internet users, irrespective of size or status (companies or individuals) may have their content cached. This openness of caches brings about severe security concerns [12, 33, 32, 26]. Among the various threats, a crucial issue in ICN is resilience to denial of service attacks performed via injection of corrupted or poisoned content in network caches [18]. In principle, a digital signature associated to every chunk [21, 30] would easily permit each ICN node to verify content integrity and provenance, and hence would permit network caches to store only *valid* objects. In practice, even neglecting the non trivial issue [19, 5] of how to gather and handle a trusted public key for verification (see section 2.1 for further details), signature verification at wire-speed appears hardly at reach.

Indeed, stretching existing state of the art signatures (and their computationally expensive cryptographic primitives, such as modular exponentiations or elliptic curve point multiplications) so as to perform wire-speed verification over line-rates in the order of 10 or 100 Gbps is hard, to say the least. Even with significant optimizations, [24] could not attain more than 3k RSA-1024 decrypts per second over a 3 GHz Intel i7 processor. [18] was bounded to a throughput of ∼150 Mbps for verifying 1500 bytes signed packets with an Intel Core 2 Duo 2.53 GHz CPU, despite using the most convenient RSA public exponent - namely 3. And Identity-based signatures, which bring about the notable advantage of avoiding the delivery of public key certificates along with the data chunk [34], incur in a verification speed consistently slower than RSA, even considering more recent lightweight approaches [17] based on Schnorr-like constructions (rather than on much more expensive pairing schemes).

True, Internet appliances exploiting extensive hardware acceleration may significantly raise the performance bar. Tilera's 100 cores single chip processor with public key accelerator enables up to 50,000 RSA handshakes per second[2], and high end security processors such as Nitrox III[3] may

[1]Hereafter, we use the terms "objects", "items", and "chunks" as synonyms; in other words we do not differentiate between independent data items versus segments of a streamed content, as long as such segments can be independently addressed (such as in emerging ICN systems, or in existing streaming solutions such as the proprietary Apple's HTTP live streaming or the MPEG-DASH standard).

[2]http://www.tilera.com/about_tilera/press-releases/tilera-announces-worlds-first-100-core-processor

[3]http://www.cavium.com/pdfFiles/NITROX-III_PB_Rev1.0.pdf?x=3

reach a trailblazing figure of up to 200k RSA ops/s.

Still, HW acceleration brings about significant extra costs, and deployment of dedicated hardware devices may clash with the emerging trend of virtualizing network functions [13], namely delivering software-based network functions running on commodity hardware.

Contribution

This paper revolves around a foundational question: *what do we lose if we cannot afford signature verification at line speed*? Apparently, we either lose in security, by accepting to cache unverified content, or in performance, by caching only the fraction of data which can be processed for signature check (and successfully verified). Goal of this paper is to show that such latter "lossy caching" operation may not necessarily be a shortcoming!

More specifically, we refer to the case of a Least Recently Used (LRU) cache not storing *every* object being retrieved (as normally expected), but only a potentially small subset of (randomly sampled) items that an *overloaded* signature verification engine is able to process. In this framework, our contribution is twofold:

First, we devise a **new analytical model** based on (very) elementary renewal arguments which permits us to capture (also) the above "lossy" caching scenario, under quite general assumptions on the distribution of the inter-arrival time among consecutive requests for a same item. We remark that the analysis of ordinary (non "lossy") LRU caches is a special case of our proposed framework, and thus we contribute in improving the state of the art also in this area. Indeed, we extend the approach first proposed in [9], and recently revisited in [16, 3], to handle a quite general arrival model suited to account for the presence of *temporal locality* in the request stream [4, 15], and hence beyond the *Independent Reference Model* used in several related works [9, 16, 31, 22, 11].

Second, our numerical investigation, comprising analytical results, simulation, and real trace analyses, appear to suggest that **the inability to cache *all* the possible content has only mild implications on performance**. Actually, in some specific (but practical) conditions, we show that the need to restrict caching to a subset of traffic may even *turn out as an advantage*.

Finally, we are of course not the first to address "probabilistic" caching: randomized LRU extensions are widely discussed since at least [31, 22]. Rather, our work distinguishes in terms of modeling assumptions (we do not restrict to the Independent Reference Model), as well as scope and motivation: the probabilistic operation and the inability to cache all possible content comes as a *non controllable aspect* (slow content integrity verification), rather than as a tunable feature related to content size or other content access costs [31, 22], or as a feature leveraged (eventually with additional per-object information) in multi-level caching systems [25, 29, 8].

2. SYSTEM MODEL

2.1 Scenario

The scenario addressed in this paper is highlighted in Fig. 1. We assume that the cache is managed with the Least Recently Used (LRU) replacement policy, being it widely exploited in practical scenarios. We recall that LRU is a simple and scalable policy, which assumes that *every cacheable content is stored*, irrespective of any measured statistics (e.g. unlike other more sophisticated strategies). We further assume that the cache is entitled to store only "valid" objects, where validity is checked by verifying a digital signature field in the data chunk. We further assume that content objects are self-contained for what concerns verification needs, i.e. that the (certified) public key needed to verify the signature is either transmitted along with the data chunk (efficient low-overhead schemes do indeed exist, e.g. an ECDSA-P192/SHA-1 signature *plus* its ECQV "implicit" certificate [1] requires in total only 104 bytes, 90% smaller than RSA), or Identity Based signatures are employed [17, 34], such that the object name itself acts as public key for verification purposes.

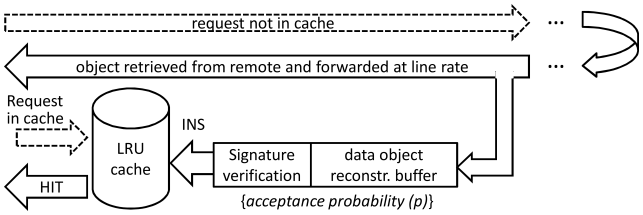

Figure 1: Scenario - content verification *not* integrated in the cache, nor uses any content statistics

As illustrated in Fig. 1 we assume, on purpose, that the signature verification task is decoupled from the cache operation, and specifically that content information gathered from the wire is sent to a front-end verification module, whose output (if the verification is successful) is then provided to the cache. Thus, deployment can occur also on legacy caching systems. Specifically, we take the *worst-case* assumption that no specific criterion (e.g. based on content types, provenance, statistics, etc) is used in selecting which content items shall be captured by the link for verification purposes. Rather, we assume that a content item is opportunistically captured by the link whenever a (small, e.g. just for the purpose of reconstructing a data item) buffer space becomes available in the verification module, i.e. right after the termination of a verification task.

It readily follows that only a subset of content chunks will be actually delivered to the cache; for example, an (optimistic) 0.1 ms signature verification time necessarily tops at 10 Kchunk/s, whereas an half loaded 10 Gbps core network link may deliver more than 15 times such traffic, even considering relatively large 4 KB chunks; in other words, in such example settings, only approximately 1 out of 15 chunks could be actually verified *before* being cached.

2.2 Model assumptions, notation, background

It is well known from traditional (e.g., NetFlow) traffic sampling, that under mild assumptions (large amount of flows, packets sufficiently interleaved, sampling rate small, see a rigorous analysis in [7]), deterministic sampling is practically equivalent to probabilistic sampling. Moreover, as we will discuss below, our model is built from the point of view of *each* single stream of requests for a same item, and thus it appears reasonable to summarize with a given *acceptance probability p* the probability that a content item, not included in the cache and forwarded by the node, is selected for signature verification and subsequently inserted in

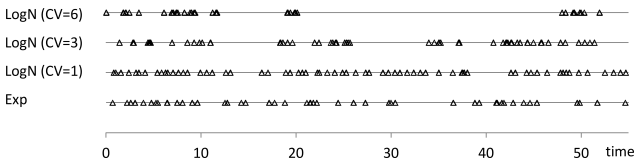

Figure 2: Example arrival patterns: Exponential and Lognormal distribution with coefficients of variation CV=1,3,6 ($CV = \sigma/m$), and mean inter-arrival time $m = 1$.

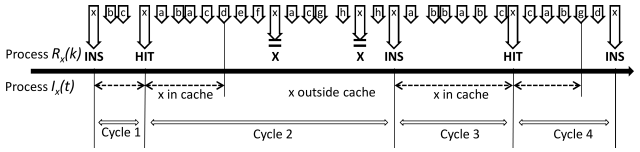

Figure 3: Request arrivals for object x; underlying renewal structure; examples for both the discrete-time process $R_x(.) = \{\text{INS}, \text{HIT}, \text{X}, \text{X}, \text{INS}, \text{HIT}, \text{INS}\}$ and the continuous-time process $I_x(t)$.

the cache (generalization to different acceptance probabilities per item being trivial). The special case $p = 1$, i.e., all items cached, corresponds to the ordinary LRU operation.

Concerning the cache system, we assume, for modeling convenience, that the storage capacity C is expressed as number of objects that may be stored therein. This assumption is practical in case of in-network caching, where the cache capacity may be bounded by the size of lookup table used to index the stored items. Otherwise, this assumption implies items of same size; extensions to uneven sizes can be addressed, e.g. as discussed in [9, 16].

We assume that objects are drawn from an universe size of (large) cardinality N. Objects are conveniently named using the index x, with $x \in \{1, 2, ..., N\}$. Each object is characterized by an associated *average arrival rate* λ_x. We assume stationary arrivals (non stationary arrivals have been very recently considered in [3] under the assumption of Cox arrivals), and consequent long-term content popularity distribution $q_x = \lambda_x / \sum_{i=1}^{N} \lambda_i$ which, unless otherwise specified, we quantify with a Zipf (non restrictive, as our model does not require to specify such distribution).

In terms of traffic arrivals, we *do not rely on the Independent Reference Model*, frequently assumed in related analytical works [9, 16, 31, 22, 11]. Rather, we model the system under more general conditions, by assuming that the inter-arrival times between two consecutive requests for a same item are independent and identically distributed random variables with general probability distribution function $F_x(t)$. Even if the extension to semi-Markov processes is work in progress, the i.i.d case appears already sufficiently descriptive to capture a wide range of *temporal locality* conditions and practical bursty-like traffic patterns (see illustrative example in Fig. 2).

Che's approximation [9]

Our model extends and casts to a more general setting a clever approximation originally introduced in [9] by Che et al. Let us focus on an object x. In most generality, the time elapsing between the instant of time the object is inserted (refreshed) in the cache, and the instant of time the object is evicted from the cache (under the assumption than no other requests for the same object arrive to the cache in the mean time), is a random variable with non trivial (and a priori unknown) distribution. [9] suggests that, for practical (reasonably large) cache sizes and population of objects (so that the request rate for each given object becomes a small percentage of the overall traffic), this random variable can be *approximated with a constant*, further *independent* of the specific object x considered. Despite its simplicity, such an approximation is shown in the original work [9] to yield an impressive accuracy. A closer look to the reasons and explanations behind such a remarkable accuracy has been recently made in [16]. We will refer to this constant cache eviction time as t_c. Note that this constant is *not* known in advance, hence it is up to the performance model to (further) determine its actual value.

2.3 Analytical Model

Intuitively, our proposed modeling approach is best understood by looking at the descriptive example of Fig. 3. We model the process of arrival of requests to the cache from the point of view of a specific object $x \in (1, N)$. A request for x arriving to the cache can have one among three possible outcomes:

- HIT: the request is served by the (*cache hit*); this implies that the object x has remained in the cache since the last arrival;
- INS: the object x is not found in the cache and thus is downloaded from a remote server; while downloaded, the object is also *inserted* in the cache;
- X: the object x is not in the cache, but differently from INS, it is *not* stored in the cache while being downloaded (i.e., verification buffer full).

The first two cases are usual according to the LRU operation. The third case comes into play when considering that the signature verification process acts as bottleneck to the cache insertion operation, and thus an arriving object may not be included in the cache upon arrival, as the signature verification buffer is found full. Per our assumption, this occurs with probability $1 - p$ independent at each subsequent object.

Owing to the LRU policy, whenever an object is either inserted in the cache or refreshed by a cache hit, its status is renewed, e.g. the age since the previous access is reset. Therefore, under the assumption of i.i.d. request inter-arrival times, the instants of time marked in the figure as HIT and INS can be conveniently employed as renewal points for *some* convenient processes, formally introduced below, and describing the evolution of the object x under consideration. In particular, from Fig. 3 we note that two different types of *cycles* may occur: i) cycles 1 and 3 end up with a cache HIT, and thus are characterized by the fact that the object x remains in the cache throughout the whole cycle itself; ii) cycles 2 and 4 instead show that after some time (number of arrivals of other distinct objects), x is evicted from the cache; unlike an ordinary LRU scheme it is not necessarily reinserted at the first next arrival (although this is clearly possible, see e.g. cycle 4), but reinsertion may happen after a number of requests (e.g. only the third request in cycle 2 gets reinserted).

Cache hit probability

Let us model the arrival of subsequent requests for a same *tagged* object x with a discrete-time process $R_x(k)$, which summarizes into one of the three possible states $\{\text{HIT}, \text{INS}, \text{X}\}$ the outcome associated to the k-th request arrival. As discussed above, the time instants $\{k|R_x(k) = \text{INS} \vee R_x(k) = \text{HIT}\}$ are renewal points for the process $R_x(k)$.

Let us now define with H_x the (so far still unknown) probability that object x is *not* evicted from the cache during a cycle. From elementary renewal theory, the steady-state hit probability $P_{hit}(x)$ for the tagged object is computed as

$$P_{hit}(x) = \frac{E[\text{no. hits per cycle}]}{E[\text{no. reqs per cycle}]} = \frac{H_x}{H_x + (1 - H_x)/p}. \quad (1)$$

The numerator is trivial, as we either have no hits in a cycle, or the cycle ends with an hit event, with probability H_x by our own definition. To compute the denominator, we observe that we have only two possible cases: i) if the cycle ends with an hit (with probability H_x), then it includes only one request; conversely ii) if the cycle envisions the eviction from the cache of the object x, which occurs with probability $1 - H_x$, then the cycle comprises on average $1/p$ requests, being p the (Bernoulli) probability that a related content is captured by the signature verification module and hence admitted to the cache[4]. By averaging over all the population, the total cache hit probability is given by:

$$P_{hit} = \frac{\sum_{x=1}^{N} \lambda_x P_{hit}(x)}{\sum_{i=1}^{N} \lambda_i} = \sum_{x=1}^{N} q_x P_{hit}(x). \quad (2)$$

Equation (1) depends on N unknown parameters H_x, a different one per each object. Che's approximation [9], discussed in the previous section, can now be very effectively exploited to reduce the number of parameters to as little as a single unknown. Indeed, [9] suggests to approximate the cache eviction time *as seen by any object* with the *same constant* t_c. As shown in [16], this appears reasonable with large population and cache size. With such approximation, being $F_x(t)$ the cumulative probability distribution function of the inter-arrival time between two consecutive requests for a same object x, we trivially conclude that

$$H_x = F_x(t_c) \quad (3)$$

as the probability H_x that the object x is not evicted is equivalent to the probability $F_x(t_c)$ that the next arrival occurs after at time interval smaller than t_c. We remark that every object x may have a different request inter-arrival distribution $F_x(.)$, but those distributions now become *given parameters* of the model, whereas the only remaining *unknown* is now just t_c, derived in the next step.

Cache eviction time

Unfortunately, our more general (non Poisson) traffic assumptions do not permit us to derive the parameter t_c as in [9]. We thus resort (again) to a renewal argument applied to a tagged object x, but this time we consider a *different*, continuous-time, *Indicator* process $I_x(t)$ which is equal to 1 when the object x is stored in the cache, and 0 otherwise.

Again, Fig. 3 shows that the process $I_x(t)$ exhibits the same, very convenient, renewal structure identified before for $R_x(k)$, being renewal points the occurrence of either cache hits *or* insertions; the crucial difference is that now the time scale is *continuous*, and thus the cycle length is measured in time units (e.g., seconds). It readily follows, again from the elementary renewal theorem, that

$$E[I_x(t)] = \frac{E[\text{time spent in cache per cycle}]}{E[\text{duration of a cycle}]} =$$

$$= \frac{\int_0^{t_c} (1 - F_x(t))\, \mathrm{d}t}{\frac{1}{\lambda_x}\left[F_x(t_c) + (1 - F_x(t_c))/p\right]}. \quad (4)$$

The denominator is the length of a renewal cycle measured now in time, hence the average inter-arrival time $1/\lambda_x$ among subsequent requests, multiplied by the average number of requests comprising a cycle (denominator of (1), having already substituted (3)). The numerator is the expected value of the random variable defined by $\min(T_x, t_c)$, where T_x is the (r.v.) inter-arrival time between requests having the usual CDF $F_x(.)$, and t_c is the constant cache eviction time. Indeed, the time spent in the cache in a considered cycle is either the inter-arrival time of the next request for x, if this comes before the eviction time t_c, or it is bounded by t_c.

Since, at each time instant, the cache contains exactly C objects, it follows that

$$\sum_{x=1}^{N} I_x(t) = C \quad \rightarrow \quad \sum_{x=1}^{N} E[I_x(t)] = C \quad (5)$$

where the obvious dependence among the processes $I_x(t)$ for different objects x is not a concern when taking expectations. By substituting (4) into (5) we finally obtain an equation whose numerical solution permits to derive the value t_c to be used in (1) and (2) for computing the relevant cache hit probabilities.

3. PERFORMANCE ANALYSIS

In this section we discuss the performance implications that emerge as a consequence of the need to restrict caching to a fraction p of requests. All analytical results presented in what follows are backed up by simulation results.

Unless otherwise specified, results are obtained using a cache size $C = 1000$ and a total population of 10^6 content items. Each item x is characterized by a popularity q_x drawn from a Zipf distribution with slope coefficient $\alpha = 0.8$. We remark that our model does not restrict to specific popularity distributions, but takes as input the (general) distribution of the inter-arrival time between consecutive requests for a same item. Although in principle request interarrivals per different items may follow different probability distributions, for convenience we report results only for homogeneous distributions with frequency of requests proportional to the popularity q_x. Specifically, we consider Exponential inter-arrivals, to model the case of a request stream that follows the independent reference model (IRM) [16], and Lognormal or Hyperexponential distributions reproducing the case of a request stream with temporal locality. In these latter cases, following [14], we change the *coefficient of variation* (CV, defined as the ratio between the standard deviation and the mean value) to affect the temporal locality. As graphically shown in Fig. 2, for a same mean inter-arrival time, the larger the CV, the burstier the request arrival process.

[4]for our performance analysis purposes, we non restrictively assume that all the object verifications are successful.

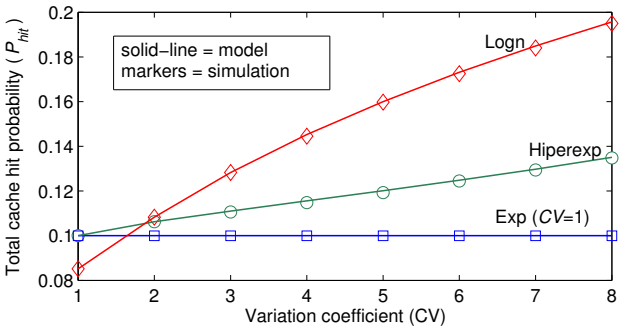

Figure 4: Total cache hit probability vs variation coefficient with $p = 1$

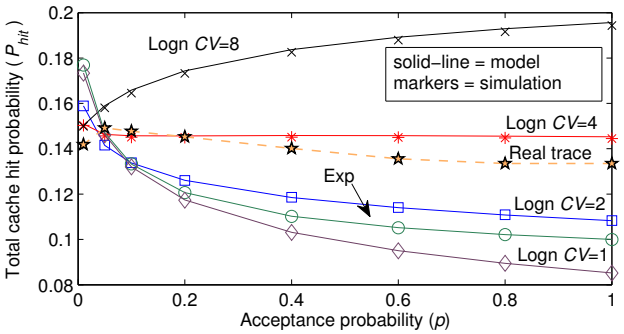

Figure 5: Total cache hit probability vs acceptance probability for Exponential, Lognormal and real inter-arrivals

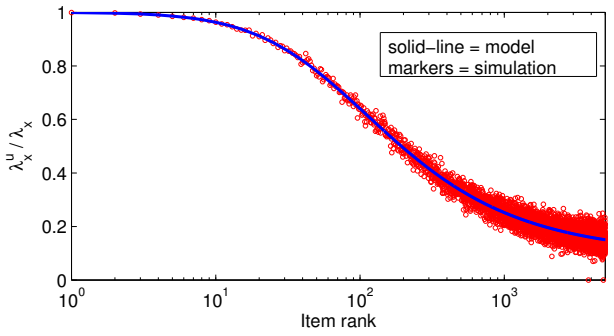

Figure 6: Update rate over request rate vs item rank for Lognormal inter-arrivals with $CV = 4$ and $p = 0.1$

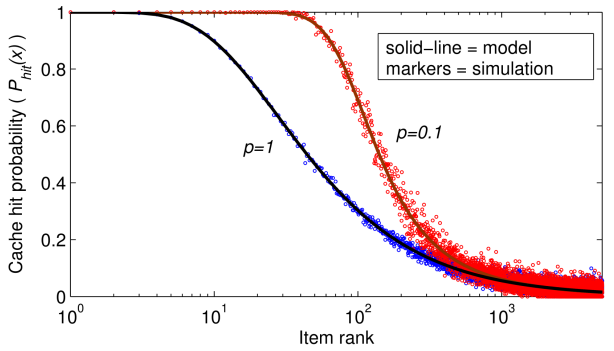

Figure 7: Per-item cache hit probability vs item rank for Exponential inter-arrivals (IRM)

Performance of ordinary LRU caches

The analytical model introduced in section 2.3 relies on a parameter p called *acceptance probability*. By setting $p = 1$ we can thus quantify performance also in ordinary LRU caches, for general distribution of the request inter arrival time. Ordinary LRU caches have been extensively studied in the literature, and it is well known (see e.g. [14]) that an increased temporal locality improves cache performance. As expected, this finding is confirmed by the results presented in Fig. 4. Here, results are shown for three different distributions of the request interarrival time (Exponential, Hyper-Exponential, and LogNormal). For the case of Hyper-Exponential and LogNormal, different coefficients of variations CV, ranging from 1 to 8, are plotted.

Fig. 4 shows that caching performance, measured in terms of cache hit probability, improve for a greater CV (more than doubling in the LogNormal case for CV going from 1 to 8). Moreover (as our model further clarifies) performance significantly depend on the chosen inter-request distribution. Indeed, Fig. 4 shows that the Exponential distribution ($CV = 1$) provides better performance with respect to those obtained by a Lognormal distribution with $CV = 1$, and shows a quite significant difference between Lognormal and Hyperexponential distributions for a same CV.

Finally, Fig. 4 shows a remarkable agreement between analytical (solid lines) and simulation results (markers).

Impact of acceptance probability p

In this section we explore the impact of the acceptance probability on the performance of a single cache. Analytical (solid lines) and simulation (markers) performance for Exponential inter arrival times, as well as for Longnormal with different coefficient of variation CV are reported in Fig. 5. Unlike the case of ordinary LRU caches ($p = 1$), Fig. 5 shows that performance, measured in terms of the resulting cache hit probability, do not necessarily increase with an increased temporal locality, quantified through the parameter CV. Indeed, the reduction of the acceptance probability p may be either *beneficial or detrimental* in terms of the resulting cache hit probability, depending on the *amount of temporal locality of the considered request stream*.

When the temporal locality is low, for instance in the case of Exponential inter arrival distribution (Independent Reference Model), or in the case of LogNormal distribution with $CV = 1$, we notice an increase of the hit probability while reducing p. Conversely, in presence of a strong temporal locality, e.g. Lognormal with $CV = 8$, hit probability tends to decrease. These results suggest that temporal locality plays a crucial *qualitative* role in the performance accomplished by caching only a subset of traffic requests, and specifically that the practical inability to cryptographically verify all cached content can even turn into an *advantage*, when temporal locality is (relatively) low.

An explanation

When p is lower than 1, not all requests can be cached, but only a randomly sampled fraction p of them are. In other words, a small p yields a reduction of the rate at which items do *update* the cache status with respect to the actual request arrival rate: whereas in an ordinary LRU cache a request for an item i guarantees that the item will be inserted in the cache, in our scenario this is conditioned to the probability p that such a request can be inserted in the signature verification buffer. Furthermore, and most interesting, such a reduction is not balanced across items, but strongly depends on the items' popularity. This is quantified by Fig. 6, which plots, for the case $p = 0.1$, the *update ratio* (the fraction of requests for a given item that are either found in the cache - hence refreshed - or inserted in the cache) versus the item index.

Indeed, the rate of arrival of requests for an item x that actually update the cache (HIT or INS) is related to the arrival rate of requests for the same item by the relation $\lambda_x^u = \lambda_x(P_{hit}(x)+(1-P_{hit}(x))p) = \lambda_x/\left(F_x(t_c)+(1-F_x(t_c))/p\right)$. It follows that popular items ($F_x(t_c)\to 1$) will have $\lambda_x^u \to \lambda_x$, whereas for infrequent items ($F_x(t_c)\to 0$), $\lambda_x^u \to p\lambda_x$.

Now, it is very interesting to note that such an uneven reduction of the cache update rate per item may improve or worsen the cache hit probability, depending on the amount of temporal locality.

Low temporal locality - If the temporal locality is low, as for instance in the Exponential (IRM) inter-arrival case, as also pointed out in [22], sampling a subset of requests yields performance improvements (see Fig. 5) since it promotes the caching of more popular items. This behavior is confirmed by Fig. 7, which shows the per-item cache hit probability in the case of Exponential inter-arrival times. We observe that a reduced acceptance probability $p = 0.1$ amplifies the hit probability for most popular contents, with respect to the ordinary LRU case ($p = 1$). This change of the cache hit probability distribution compensates the sub-optimality of the LRU caching policy and makes cache performance closer to an ideal replacement algorithm called A0, which tends to store just the most popular items, and which is optimal under the IRM assumption [27, 10].

High temporal locality - The situation completely changes as temporal locality gets larger, as for instance in the Lognormal inter-arrival case with $CV = 8$ (see Fig. 5). Indeed, a reduction of the update rate, i.e. of p, implies an "inertia" in the refresh of cache contents. The inertia reduces the ability of LRU to follow the temporary (but harsh) changes of popularity, that characterize a request stream with strong temporal locality, and this could cause an overall decrease of the hit probability. An evidence of the "inertia" is provided by Fig. 8 that reports the average *cache period* of a content object, since the insertion time of the item in the cache up to its removal time, i.e. $\int_0^{t_c}(1-F_x(t))\,\mathrm{d}t/(1-F_x(t_c))$. As expected, higher ranked objects report an higher cache period and, in addition, we observe that the reduction of acceptance probability p implies an increase of such period, i.e. of cache inertia.

Medium temporal locality - In medium conditions, the benefit of promoting caching of popular items tends to balance the disadvantage due to the increasing cache inertia. Focusing on results of Fig. 5, in case of Lognormal with $CV = 4$ the balance is fair and we have a rather flat performance versus p.

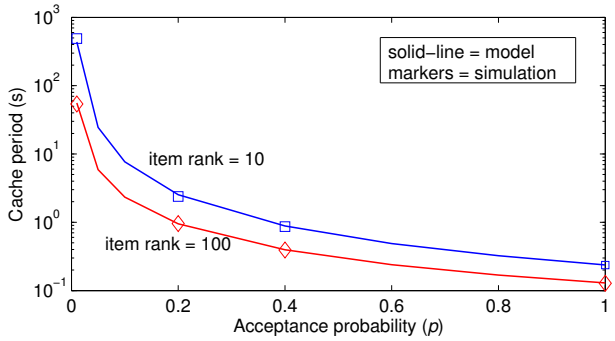

Figure 8: Average item cache period vs acceptance probability for Lognormal inter-arrivals with $CV = 4$

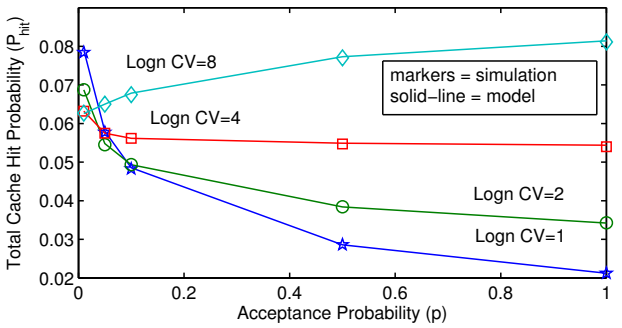

Figure 9: Total cache hit probability vs p for Lognormal inter-arrivals; cache size: 100 items

Impact of assumptions and parameters

A first natural question is whether the above findings are confirmed for different cache sizes. In Fig. 9, we have re-run all results presented in Fig. 5 for the LogNormal case, but this time using an extremely small cache size of just $C = 100$ items (i.e. 0.01%, instead of 0.1%, of the entire considered set of items). Clearly, quantitative results are significantly different (the cache hit probability is much smaller, as expected). However, the performance trends varying the parameters p and CV are confirmed.

A second remark concerns the approximation of the system operation through a single (bernoulli) acceptance probability p. In practice, requests will be queued in a signature verification buffer. To assess the impact of such an approximation, Fig. 10 reports simulation results where the system operation is modelled using a FIFO verification buffer of size 100 items. Via simulation, we measure the actual *average* acceptance probability (hence not anymore a bernoulli process), and we use this measured average value as x-axis quantity for plotting the simulation results. The figure shows that such results (markers) still practically lie on the analytical curve (solid lines), thus implying that the verification buffer dynamics do not meaningfully affect performance.

4. REAL TRACES AND CACHE NETWORKS

In order to confirm whether our findings hold also in real world conditions, we have run selected experiments using real world traces made available by the IRCache project (www.ircache.net), with specific reference to data gathered

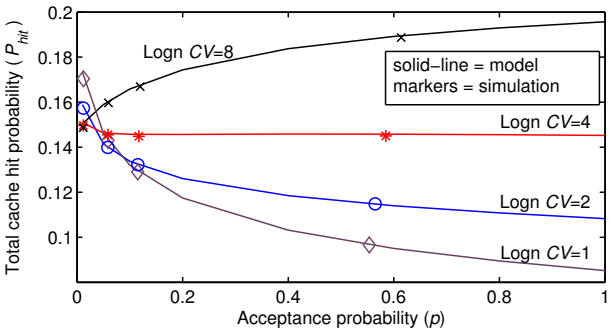

Figure 10: Total cache hit probability vs acceptance probability for Lognormal inter-arrivals and a FIFO verification queue of length 100 items

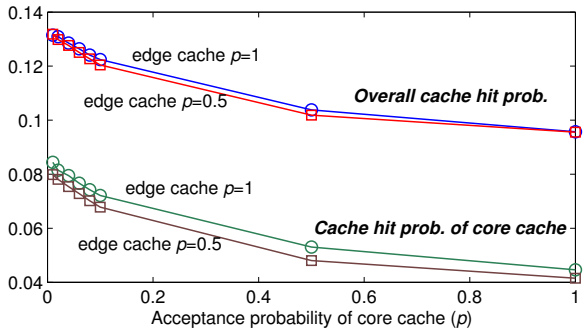

Figure 11: Cache hit probability of cache network vs acceptance prob. p of core cache

Table 1: Traces parameters

Date	# Obj	# Req	1-timers	CV	α
02/18	854241	3571125	68.30%	3.1439	0.8165
02/19	993711	4121865	68.66%	2.8393	0.8152
02/20	871565	3593373	69.27%	2.2542	0.8138
02/21	811827	3416817	67.61%	2.0523	0.8211

from the SD Network Proxy (the most loaded proxy to which end-users can connect) in February 2013. A detailed inspection shows that such traces capture regional traffic, and as such exhibit significant non stationarities due to daily traffic fluctuations. Therefore, to measure the performance expected in the busy hour, we considered only traffic traces taken in 4 hours peak traffic periods. Table 1 reports some parameters of the trace, including the percentage of one-timers (requests arriving only once during the trace lifetime), the average[5] coefficient of variation CV, and the α parameter of the best approximating Zipf popularity distribution, computed via the curve-fitting tool from [6].

Results for a single cache are shown in Fig. 5, using as input data a trace gathered on February 18, 2013. For $p > 0.05$, performance slightly improve as p reduces (following a trend somewhat intermediate between the case of Lognormal with $CV = 2$ and $CV = 4$ - we remind that the trace exhibits a mean CV of the inter arrival time of 3.14). Below $p = 0.05$, performance instead slightly reduce.

Of greater practical interest is the understanding of what can happen in a scenario envisioning a network of caches (the actual scenario of an Information Centric Network) rather than just a single cache. Since the extension of the analysis to such setting is not immediate (currently work in progress), we preliminary assess the implications of a cache network using simulations feeded by real traffic traces.

Our interest for cache networks stems from the fact that, as pointed out in [14], the stream of requests missed by edge caches exhibits a reduced temporal locality (compared to the original request streams), which in turns reduces the effectiveness of core network caches. Now, in realistic scenarios, wire-speed content integrity verification may be feasible in edge caches, but it is definitely an hard task in core network caches where the forwarding rates may easily exceed tens of Gbps (in other words, the acceptance probability p as defined in this paper will be necessarily small, say in the order of 10^{-1} or 10^{-2}). But if, as suggested in [14], temporal locality in core network caches is smoothed by the "filtering" effect of edge caches, *the inability to verify content on the fly may actually improve caching performance in the network core*!

To preliminary assess this intuition, we have run a simulation in an basic cache hierarchy scenario, formed by a core cache that serves 4 edge caches. The four edge caches are loaded with real traces of the same four hours of four consecutive days; i.e. the trace of February 18+i 2013 loads the i-th edge cache, with $i = 0..3$. The edge cache size is 100 items and the core cache size is 1000 items (qualitatively similar results are obtained also for edge caches of size 1000, not reported).

Fig. 11 shows the overall cache hit probability of the cache network and the cache hit probability of the core cache, versus the acceptance probability of the core cache. We consider two cases of p for the edge caches, namely $p = 1$ and $p = 0.5$. We observe that a reduced p in the core cache yields a *benefit* in terms of overall cache hit probability, due to the advantage of promoting the caching of popular items, as previously discussed. The same behavior occurs for the single core cache. Moreover, we observe that reducing the acceptance probability on the edge from 1 up to 0.5 slightly decreases the performance of the core cache. This occurs since a greater temporal locality is reported in ingress of the core cache, so the disadvantage of the cache inertia increases.

5. CONCLUSIONS AND FUTURE WORK

Besides the theoretical contribution of a new analytical model, the main take-home message in this paper is that the ability to forward to the cache only a subset of (integrity verified) incoming items may have either beneficial or detrimental impact on the resulting cache hit probability, the ultimate balance depending on the amount of temporal locality of the considered request stream. What appears interesting is that, in practical scenarios, core network caches (where integrity verification is significantly slower than line rates) are expected to be loaded by *favorable* temporal locality streams. Work in progress aims at extending our theoretical insights to hierarchical caches, so far preliminary tackled only via real world traces and simulation.

[5]The CV value reported in the table is averaged over all the requests; a closer look to the trace shows that different items experience different inter arrival distribution (characterization not being straightforward given the small number of requests for less frequent items).

6. ACKNOWLEDGEMENTS

This work is supported in part by the European Commission in the context of the FP7/NICT EU-JAPAN GreenICN project.

7. REFERENCES

[1] Standards for efficient cryptography group (SECG), SEC4 Elliptic Curve Qu-Vanstone implicit Certificate Scheme (ECQV). v1.0, January 2013.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of Information-Centric Networking. *IEEE Commun. Mag.*, 50(7):26–36, July 2012.

[3] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, and S. Niccolini. Analyzing the Performance of LRU Caches under Non-Stationary Traffic Patterns. *ArXiv, abs/1301.4909*, 2013.

[4] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *SIGMETRICS*, 1998.

[5] M. Baugher, B. Davie, A. Narayanan, and D. Oran. Self-verifying names for read-only named data. *1st IEEE INFOCOM Nomen Workshop*, 2012.

[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *IEEE INFOCOM*, pages 126–134, 1999.

[7] Y. Chabchoub, C. Fricker, F. Guillemin, and P. Robert. Deterministic versus probabilistic packet sampling in the internet. In *20th international teletraffic conference, ITC20*, pages 678–689, 2007.

[8] W. K. Chai, D. He, I. Psaras, and G. Pavlou. Cache "less for more" in information-centric networks. In *11th IFIP TC6 conf. on Networking*, pages 27–40, 2012.

[9] H. Che, Y. Tung, and Z. Wang. Hierarchical Web caching systems: modeling, design and experimental results. *IEEE J. on Sel. Areas in Commun.*, 20(7):1305–1314, Sept. 2002.

[10] E. G. Coffman, Jr. and P. J. Denning. *Operating Systems Theory*. Prentice Hall Professional Technical Reference, 1973.

[11] A. Dan and D. F. Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. In *ACM SIGMETRICS*, pages 143–152, 1990.

[12] L. Deng, Y. Gao, Y. Chen, and A. Kuzmanovic. Pollution attacks and defenses for internet caching systems. *Comput. Netw.*, 52(5), Apr. 2008.

[13] ETSI. Network functions virtualisation - white paper. portal.etsi.org/NFV/NFV_White_Paper.pdf oct. 2012.

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

[15] R. C. Fonseca, V. A. F. Almeida, and M. Crovella. Locality in a web of streams. *Commun. ACM*, 48(1):82–88, 2005.

[16] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for LRU cache performance. In *24th international teletraffic conference, ITC24*, pages 1–8, 2012.

[17] D. Galindo and F. Garcia. A Schnorr-like lightweight Identity-Based Signature scheme. *AFRICACRYPT 2009, LNCS 5580*, pages 135–148, 2009.

[18] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS and DDoS in Named-Data Networking. *ArXiv, abs/1208.0952*, 2012.

[19] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *1st ACM SIGCOMM ICN workshop*, 2011.

[20] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-Centric Networking: seeing the forest for the trees. In *10th ACM SIGCOMM HotNets*, 2011.

[21] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking Named Content. In *5th ACM CoNext*, 2009.

[22] P. R. Jelenković and A. Radovanović. Optimizing LRU Caching for Variable Document Sizes. *Comb. Probab. Comput.*, 13(4–5):627–643, 2004.

[23] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4):181–192, 2007.

[24] M. E. Kounavis, X. Kang, K. Grewal, M. Eszenyi, S. Gueron, and D. Durham. Encrypting the internet. *ACM SIGCOMM Computer Commun. Review*, 40(4):135–146, 2010.

[25] N. Laoutaris, H. Che, and I. Stavrakakis. The LCD interconnection of LRU caches and its analysis. *Perf. Eval.*, 63(7):609–634, 2006.

[26] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda. Privacy risks in named data networking: what is the cost of performance? *SIGCOMM Comput. Commun. Rev.*, 42(5):54–57, Sept. 2012.

[27] E. J. O'Neil, P. E. O'Neil, and G. Weikum. The LRU-K page replacement algorithm for database disk buffering. In *ACM SIGMOD*, pages 297–306, 1993.

[28] G. Pallis and A. Vakali. Insight and perspectives for Content Delivery Networks. *Commun. ACM*, 49(1):101–106, Jan. 2006.

[29] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *2nd SIGCOMM ICN workshop*, pages 55–60, 2012.

[30] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi. Transport-layer issues in Information Centric Networks. In *2nd ACM SIGCOMM ICN workshop*, 2012.

[31] D. Starobinski and D. Tse. Probabilistic methods for web caching. *Perf. Eval.*, 46(2-3):125–137, 2001.

[32] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp. Backscatter from the data plane – threats to stability and security in information-centric networking. *ArXiv abs/1205.4778*, 2012.

[33] M. Xie, I. Widjaja, and H. Wang. Enhancing cache robustness for content-centric networking. In *IEEE INFOCOM*, pages 2426 –2434, march 2012.

[34] X. Zhang, K. Chang, H. Xiong, Y. Wen, G. Shi, and G. Wang. Towards name-based trust and security for content-centric network. *IEEE ICNP*, 2011.