POLITECNICO DI TORINO
Repository ISTITUZIONALE

On the Effectiveness of the PIT in Reducing Upstream Demand in an NDN Router

Original
On the Effectiveness of the PIT in Reducing Upstream Demand in an NDN Router / Ahmad, M., Roberts, J., Leonardi, E., Movaghar, A.. - In: PERFORMANCE EVALUATION. - ISSN 0166-5316. - STAMPA. - 138:(2020). [10.1016/j.peva.2020.102081]

Availability:
This version is available at: 11583/2818192 since: 2020-05-05T10:16:07Z

Publisher:
Elsevier

Published
DOI:10.1016/j.peva.2020.102081

Terms of use:

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

Publisher copyright
Elsevier postprint/Author's Accepted Manuscript

© 2020. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/.The final authenticated version is available online at: http://dx.doi.org/10.1016/j.peva.2020.102081

(Article begins on next page)

28 November 2026

On the Effectiveness of the PIT in Reducing Upstream Demand in an NDN Router

Mahdieh Ahmadi[a,*], James Roberts[b], Emilio Leonardi[c], Ali Movaghar[a]

[a] *Sharif University of Technology, Tehran, Iran*
[b] *Telecom ParisTech, Paris, France*
[c] *Politecnico di Torino, Turin, Italy*

Abstract

The paper revisits the performance evaluation of caching in a Named Data Networking (NDN) router where the content store (CS) is supplemented by a pending interest table (PIT). The PIT aggregates requests for a given content that arrive within the download delay and thus brings an additional reduction in upstream bandwidth usage beyond that due to CS hits. We extend prior work on caching with non-zero download delay (non-ZDD) by proposing a novel mathematical framework that is more easily applicable to general traffic models and by considering alternative cache insertion policies. Specifically we evaluate the use of an LRU filter to improve CS hit rate performance in this non-ZDD context. We also consider the impact of time locality in demand due to finite content lifetimes. The models are used to quantify the impact of the PIT on upstream bandwidth reduction, demonstrating notably that this is significant only for relatively small content catalogues or high average request rate per content. We further explore how the effectiveness of the filter with finite content lifetimes depends on catalogue size and traffic intensity.

Keywords: Named Data Networking, Caching, Request Aggregation, Content Popularity

1. Introduction

The well-known proposal for a clean-slate, Named Data Networking (NDN) architecture for the future Internet [1, 2] is still under active development and pre-standardization at the IRTF. A major feature of NDN is the systematic use of in-router, line rate caching meant to significantly reduce upstream bandwidth requirements by storing local copies of popular contents. NDN routers also perform *collapsed forwarding* whereby a single content download can satisfy near simultaneous requests from multiple users. The objective of the present paper is to evaluate the effectiveness of collapsed forwarding and to understand how it depends on the traffic and popularity characteristics.

In NDN, small chunks of content in the form of Data packets are requested by name by users who emit Interest packets. If an Interest matches a content in the router Content Store (CS), the request is a hit and the content is returned directly. If the content is absent, the request is forwarded to a Pending Interest Table (PIT). If there is no match in the PIT, the request is forwarded towards a known external source and the Interest is recorded in the PIT. If the PIT already has a matching entry, the current Interest is added to the record but not forwarded. The PIT entry is removed when the content Data packet arrives from the external source after a download delay and may then be stored in the CS. The aggregation of Interests in the PIT during the download delay thus realizes collapsed forwarding. The PIT can be regarded as a supplementary meta cache that only stores names and not actual contents and thus potentially alleviates some serious challenges in realizing a CS of adequate capacity operating at line rate.

To evaluate PIT effectiveness, it is clearly necessary to forego the usual assumption that downloads occur instantaneously after a request cache miss, as if there was zero download delay (ZDD). A non-ZDD assumption is required to properly account for request aggregation. We also wish to investigate the impact on performance of time locality

*Corresponding author
Email addresses: mahmadi@ce.sharif.edu (Mahdieh Ahmadi), jim.roberts@telecom-paristech.fr (James Roberts), leonardi@polito.it (Emilio Leonardi), movaghar@sharif.edu (Ali Movaghar)

in the request process or, more specifically, of the fact that content popularity is not constant but varies in time. Our model builds on several pieces of prior work. The non-ZDD CS-PIT system was first analyzed by Dehghan *et al.* [3]. We propose an alternative approach that is computationally more efficient when requests do not follow the usual independent reference model (IRM) but are modelled using general renewal processes. Our novel approach can also be more easily extended to evaluate more advanced insertion policies that enhance hit rate performance.

We use renewal processes to model time locality illustrating its generally beneficial impact on performance compared to IRM input. The particular renewal processes considered are inspired by prior work on the analysis of ZDD systems with time locality by Garetto and co-authors in [4] and [5]. Our analysis is applied to a CS implementing the usual Least Recently Used (LRU) replacement policy and also to a CS equipped with a filter that improves hit rates by preferentially inserting the most popular contents. The particular filter we evaluate is a non-ZDD variant of the 2-LRU cache considered in [4]. We use the analysis to perform extensive numerical evaluations, whose accuracy is confirmed by simulation, to explore the effectiveness of PIT aggregation and how this depends on critical parameters characterizing system and demand.

Our main contributions are the following:

- We develop an original analytical framework to compute the hit rate and collapsed forwarding performance of the non-ZDD CS-PIT system using LRU replacement under renewal traffic.
- The analysis is extended to a 2-LRU CS-PIT system where an additional LRU meta cache filter is used to avoid caching the least popular contents.
- The accuracy of the analytical framework is demonstrated by comparison with the results of simulations in an extensive series of experiments under synthetic and trace-based workloads.
- These evaluations constitute an exhaustive investigation of how the effectiveness of PIT aggregation depends on download delay, CS capacity, traffic intensity and content catalogue size.
- The impact on CS-PIT performance of finite content lifetimes (approximating varying popularity) is illustrated through results for a particular choice of renewal input and real trace data.

The rest of the paper is organized as follows. Sec. 2 reviews related work. In Sec. 3, we introduce the principal concepts and notations. In Sec. 4, we analyse LRU and 2-LRU replacement policies applied to the CS-PIT system and derive performance metrics of interest. In Sec. 5, we evaluate the accuracy of our analysis through extensive simulations and evaluate the performance of considered policies for contents with finite lifetime. Finally, we conclude the paper in Sec. 6. To facilitate reproducibility, we have made simulation and analysis code available on GitHub [6].

2. Related Work

The literature on the modelling and analysis of caching policies is vast, as exemplified by the recent survey paper [7]. We limit the present discussion to papers that are most directly related to our work on the impact of PIT request aggregation in an NDN router and the use of a pre-filter to improve hit rates.

Our analysis is inspired by that proposed by Dehghan *et al.* in [3]. Their analysis applies to non-ZDD CS-PIT systems under the characteristic time approximation [8, 9]. The authors derive expressions for hit rates and request forwarding probabilities under a renewal traffic model for caching policies including LRU. More recently, Dai *et al.* [10] have considered a different implementation of LRU for the CS-PIT system where the LRU list is updated on request arrival rather than on content insertion after download. This approach makes it possible to apply to non-ZDD caches previous theoretical justifications of the characteristic time approximation for LRU with IRM input ([8], [9]), under the assumption that content metadata is never evicted from the LRU list between request arrival and download. However the assumption made in [10] seems somehow artificial. This is the reason why in our paper we prefer to restore the more natural assumptions made in [3]. We apply the characteristic time approximation to non-ZDD caches, and we validate the accuracy of this approximation by simulation. We develop an original, computationally efficient

mathematical framework for the non-ZDD CS-PIT system under a general renewal traffic model, notably enabling evaluation of the impact of time locality.

Our work significantly extends the model of [3] by considering a more efficient cache insertion policy than simple LRU. A number of such policies have been proposed in the literature of which the following references constitute a representative sample. Their common objective is to avoid "poluting" the cache with unpopular items that are unlikely to produce subsequent cache hits. One popular device is to insert a new content only if it has been requested one or a given number of times in a preceding fixed length time period [11, 12, 13]. An alternative insertion criterion is the presence of a given number of previous requests for the content among a certain number of most recent requests [14, 15]. The final LRU cache might be preceded by a number of, real or virtual, cache stages with contents progressively moving from one to another on request arrivals [4, 16, 17]. A simpler filter mechanism is to add new contents only with a certain, small probability that may be constant [18, 4] or content specific [19, 20]. In the present work we only consider the so-called 2-LRU cache where new contents are moved to the actual LRU on a request miss only if they are present in a filter list managed as a virtual LRU cache. This mechanism has been shown in previous work to be reasonably efficient (e.g., [4, 16]) and can be readily adapted to the considered non-ZDD, CS-PIT system.

3. System Assumptions

In this section, we introduce the principal concepts and notation. We discuss the assumptions used to perform the analysis in Sec. 4 and the evaluations in Sec. 5.

3.1. CS-PIT Interplay

Caching policies are usually analyzed under the assumption that content downloads occur immediately after a cache miss request. In practice, in an NDN router, the delay between a CS miss and the content download can be significant and in this time, one or more subsequent requests may be aggregated in the PIT. In the following, such a request is referred to as a PIT hit while any request arriving while the content is in the CS is termed a CS hit. Let $p_{hit}^{cs}(k)$ denote the probability a request for some content k is a CS hit and $\mathbb{P}(\text{PIT hit}\,|\,\text{CS miss})$ the conditional probability of a PIT hit given a CS miss. The probability of a PIT hit, $p_{hit}^{pit}(k)$, is the probability a request is a CS miss and a PIT hit which can be calculated as

$$p_{hit}^{pit}(k) = \mathbb{P}(\text{PIT hit}\,|\,\text{CS miss}) \times \mathbb{P}(\text{CS miss}).$$

Any request that is a miss at both CS and PIT is forwarded upstream so that the proportion of requests that result in a download is

$$p_{fwd}(k) = (1 - \mathbb{P}(\text{PIT hit}\,|\,\text{CS miss})) \times \mathbb{P}(\text{CS miss}) = 1 - p_{hit}^{pit}(k) - p_{hit}^{cs}(k). \tag{1}$$

The round trip download delay for a forwarded request for content k is assumed to be an independent random variable denoted D_k. In this paper, we assume that each content download request will have a response. Moreover, the PIT is not subject to any storage constraint and is therefore able to aggregate all content requests arriving during the downloaded delay.

3.2. CS Insertion and Eviction Policies

Cache performance depends on the policies used to decide if a given content should be inserted and, if so, which other content must be evicted to make room. We limit our evaluation to two variants of the well-known LRU policy. LRU eviction is simple enough for operation at line speed and is more efficient than alternatives like FIFO or Random [4]. The framework could be extended to these policies, notably by applying models introduced in [4], but we do not expect our general conclusions on the impact of demand characteristics on PIT effectiveness to change.

We first consider the classical LRU policy where all downloaded contents are systematically inserted in the CS. We then consider a more selective insertion policy, which is intended to improve hit rate performance. A content is inserted on download only if its name is present in a list that preferentially records popular items. We refer to this list and its update mechanism as a 'filter'. Many possible filter mechanisms have been proposed and analyzed in the literature, as discussed in Sec. 2. Here we suppose the CS is equipped with a filter consisting of a list of content metadata updated using LRU. New contents are only added to the CS if their metadata is present in the filter list. This insertion and eviction policy applied to a ZDD cache is called 2-LRU in [4] and has been shown to be an efficient solution. Its precise specification for the non-ZDD CS-PIT system is deferred to Sec. 4.2.

3.3. Content Popularity

Cache performance depends critically on how requests are spread over the population of distinct content items. We assume here that users request items from a total population of K constant size chunks. The request rate for a given chunk is determined by a popularity distribution $\{p_k\}$, $\sum_{1\leq k\leq K} p_k = 1$, such that, if the overall request rate is λ, the request rate for content k is $\lambda_k = \lambda p_k$. The content items are ordered such that $p_1 \geq p_2 \geq \cdots \geq p_k$ and we consider Zipf popularities,

$$p_k = k^{-\alpha} / \sum_{i=1}^{K} i^{-\alpha}. \tag{2}$$

Observations on real demand have shown that this model is a reasonable approximation for Internet content retrieval (see [21] for a recent confirmation). Zipf popularity is such that, in any finite period, the cumulative distribution of the number of requests per content exhibits power law behaviour. In particular, a typically very large proportion of contents will be requested only once (so-called "one-hit-wonders" [12, 22]) while many more are not requested at all. It is thus difficult to accurately measure and model the tail of the content popularity distribution, as discussed by Olmos and Kauffmann [23].

The Zipf representation is a simple approximation that is considered adequate for the present analysis. It is sufficiently general to explore the performance of the CS-PIT system under a range of popularity profiles as determined by the values of K and α. For $\alpha < 1$ and K large, note that ZDD LRU hit rates for a cache of capacity C depend on C/K and not separately on C and K [8].

Content popularities vary over time and to ignore this variability can lead to significant errors in predicting cache performance [24, 25]. The authors of [24] and [25] independently proposed to account for varying popularities through a so-called shot noise model. In this approach, contents appear at the instants of a stochastic process and receive requests at a rate that varies over time, eventually decreasing to zero at the end of its 'lifetime'. The analysis of this model is challenging, however [25, 26]. We adopt a more tractable model first proposed by Garetto *et al.* [5] .

In the model of [5], contents are alternately active and inactive. An active phase has an exponentially distributed duration and corresponds to the content lifetime. During its active phases, requests for content k arrive as a Poisson process of rate ν_k. The inactive phase also has an exponential distribution of mean large enough that, with high probability, any cached content is evicted before the next active phase. The content thus appears in a new incarnation in each active phase. This request arrival process is known as an interrupted Poisson process (IPP) [27]. The overall request rate for content k is

$$\lambda_k = \frac{\nu_k T_{on}}{T_{on} + T_{off}}, \tag{3}$$

where T_{on} and T_{off} are the mean durations of active (on-period) and inactive (off-period) phases.

3.4. Request Process

We suppose requests for any content occur at the epochs of a *stationary renewal process* [28]. Let t_i for $i \geq 0$ be successive request times for content k. The distribution of the inter-request intervals $X_i = t_i - t_{i-1}$ is denoted $F_k(t)$ and their density $f_k(t)$. The average request rate is then $\lambda_k = 1/\mathbb{E}[X_i]$ where $\mathbb{E}[X_i] = \int_0^\infty (1 - F_k(t))dt$. The *age* of a stationary renewal process at an arbitrary instant t is the time between t and the previous request arrival and does not depend statistically on t. Let $A(k)$ denote the age of the request process for content k. Its distribution is given by

$$\mathbb{P}(A(k) < a) = \widehat{F}_k(a) = \lambda_k \int_0^a (1 - F_k(x))dx, \text{ for } a \geq 0. \tag{4}$$

The number of requests in an arbitrary interval of length t following a request arrival (e.g., in $(t_i, t_i + t]$) is a random variable denoted N_t. We also define random variable $S_n = X_1 + \cdots + X_n$ which denotes the time until the arrival of the nth request. The distribution of S_n can be calculated as

$$\mathbb{P}(S_n \leq t) = \mathbb{P}(N_t \geq n) = F_k^{(n)}(t),$$

where $F_k^{(n)}$ is the n-fold convolution of F_k. The expectation of N_t is called the *renewal function* that we denote by $m_k(t)$ and is given by

$$m_k(t) = \mathbb{E}[N_t] = \sum_{n=1}^{\infty} \mathbb{P}(N_t \geq n) = \sum_{n=1}^{\infty} F_k^{(n)}(t). \tag{5}$$

In our evaluations we consider some particular renewal processes. The simplest is the Poisson process where $F_k(t) = 1 - e^{-\lambda_k t}$. This choice models the so-called independent reference model (IRM) where the probability an arbitrary request is for content k is independent of all previous requests and equal to p_k. The IRM ignores variations in relative popularity over time and all *temporal locality* between requests, i.e., the fact that if a content is requested at some instant in time, then the probability of a request for the same content arriving in the near future tends to increase.

As discussed in Sec. 3.3, time varying popularity can be modeled using the IPP. This is a renewal process where intervals $(t_{i+1} - t_i)$ have a hyper-exponential distribution with two states [27]. In Sec. 5 we consider a particular hyper-exponential renewal process to evaluate the accuracy of the analysis before fitting the parameters of this model to statistics derived from trace analyses.

3.5. The Characteristic Time Approximation

To evaluate CS-PIT system performance we adapt the now well-known characteristic time approximation. This approximation has become popular following its proposal by Che *et al.* [29] for evaluating LRU under the IRM, and its later analytical justification by Fricker *et al.* [9]. It was, however, first derived as an accurate asymptotic limit by Fagin in a paper from 1977 [8]. It has recently been applied more extensively to other cache insertion and eviction policies with IRM or renewal input, in [4], [30], [31] and [32].

For an LRU cache, the approximation consists in assuming a content inserted at some instant and not subsequently requested will be evicted after a deterministic characteristic time T_C. This represents the time for requests for C distinct contents to occur where C is the cache capacity. For a renewal request process, the probability an arbitrary request for content k will be a hit is then

$$p_{hit}(k) = F_k(T_C), \tag{6}$$

while the probability the content is present in the cache at an arbitrary instant is

$$p_{in}(k) = \widehat{F}_k(T_C). \tag{7}$$

T_C is determined on numerically solving the equation

$$C = \sum_{k=1}^{K} p_{in}(k). \tag{8}$$

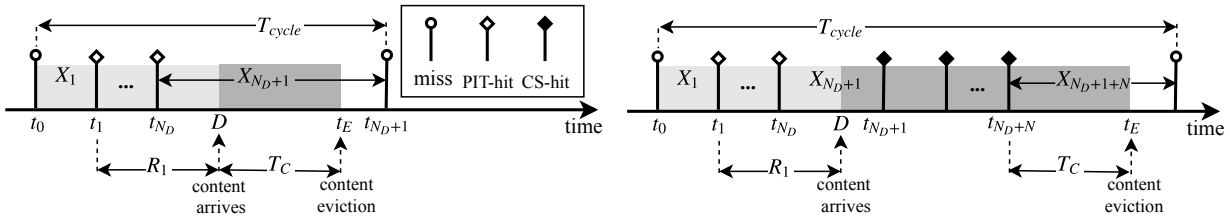
Note that under the characteristic time approximation the system is modeled as a cache which has unlimited instantaneous capacity but limited average capacity equal to C, where contents have a constant time to live (TTL) equal to T_C [4, 33]. The TTL is reset to T_C when the content is inserted and on every subsequent cache hit. This interpretation is used in the following analysis.

4. Performance of non-ZDD Policies

We derive characteristic time approximations for the hit rate performance of a CS implemented as an LRU cache or as an LRU cache with filter, accounting for non-zero download delay.

4.1. LRU CS

The LRU CS is implemented as a double linked list of pointers to stored content. Items are moved to the front of the list on insertion following a download and at the instants of subsequent requests that are hits. When a new item is inserted at the front, the last item in the list is the least recently used and is evicted. A non-ZDD LRU cache differs from classical LRU in that insertion does not occur immediately following a request miss but is deferred for a download delay D. Any further requests occurring in this delay are aggregated in the PIT.



(a) $t_E = D + T_C$ (b) $t_E = t_{N_D+N} + T_C$

Figure 1: Request process and CS status for a given content under non-ZDD LRU.

To compute hit rates, we apply the characteristic time approximation interpreting T_C as the common 'time to live'. We consider the sojourn of any item in the CS-PIT system that, in this interpretation, is independent of that of other items. For brevity we omit the index k identifying the content in question in previously introduced notation. Figure 1 illustrates occupancy cycles delimited by requests that are a miss for both CS and PIT. As requests occur as a stationary renewal process, these cycles are statistically independent and hit rate performance can be derived from expected values in a typical cycle.

The cycle begins with a request miss at time t_0 and terminates with the next miss following content eviction, where content eviction occurs at time t_E. Without loss of generality we set $t_0 = 0$. The miss at 0 triggers an upstream request (the Interest packet is forwarded towards a known source) and an initial registration in the PIT. The content is downloaded and arrives after delay D. Any requests made between 0 and D are PIT hits and are not forwarded. The number of such requests is N_D.

Requests arriving between times D and t_E are CS hits. The number of such hits is $N \geq 0$. We have $N = 0$ if the remaining inter-request interval following D is greater than T_C and $t_E = D + T_C$ (Figure 1a). For $N > 0$, the content is evicted after the first interval that is greater than T_C. In this case $t_E = t_{N_D+N} + T_C$, where t_{N_D+N} denotes the time of occurrence of the last CS hit request (Figure 1b).

The performance of this system was analysed by Dehghan *et al.* [3]. They propose a formal evaluation for a cache with a general renewal request process implementing LRU, FIFO or Random insertion policies but only present numerical results for the IRM, i.e., for Poisson requests. The derived formulas are difficult to apply in practice for more general processes and demand prohibitively long execution times (see Appendix A). We have therefore derived a simpler, novel approximation that we now describe.

To compute T_C from (8) we need an expression for p_{in}^{cs}, the probability the content is in the CS at an arbitrary instant t. This occurs if one of the following holds:

(i) the last request before t was a CS hit and arrived in $[t - T_C, t)$, or

(ii) the last request before t was not a CS hit, the content download occurred before t and the content was not evicted before t (i.e., $D \in [t - T_C, t]$).

Event (i) occurs when the age of the request process A is less than T_C and, from (4), has probability $\widehat{F_k}(T_C)$. Let R be the residual download time at the arrival time of the last request before t given that this was a CS miss. Event (ii) can then be expressed $R < A < R + T_C$. We deduce the expression

$$p_{in}^{cs} = p_{hit}^{cs} \cdot \widehat{F}(T_C) + (1 - p_{hit}^{cs}) \cdot \mathbb{P}(R < A < R + T_C). \tag{9}$$

A similar argument can be applied to deduce an expression for p_{hit}^{cs}. In this case the situation of the content in events (i) and (ii) is considered at a request instant yielding

$$p_{hit}^{cs} = p_{hit}^{cs} \cdot F(T_C) + (1 - p_{hit}^{cs}) \cdot \mathbb{P}(R < X < R + T_C), \tag{10}$$

where X represents the last inter-request interval. Solving (9) and (10), we have,

$$p_{in}^{cs} = \frac{Z\widehat{F}(T_C) + \widehat{Z}(1 - F(T_C))}{1 - F(T_C) + Z} \tag{11}$$

$$p_{hit}^{cs} = \frac{Z}{1 - F(T_C) + Z}, \tag{12}$$

where $Z = \mathbb{P}(R < X < R + T_C)$ and $\widehat{Z} = \mathbb{P}(R < A < R + T_C)$. In Sec. 4.1.1, we will explain how to approximate Z and $\widehat{Z}$ using moments of the residual download time. Finally, using Eq. (1), the forwarding probability is given by

$$p_{fwd} = \frac{(1 - p_{hit}^{cs})}{1 + \mathbb{E}[m(D)]}, \tag{13}$$

where $\mathbb{E}[m(D)]$, the expected number of PIT hit requests, is computed with respect to the distribution of download delay D and $\mathbb{P}(\text{PIT hit} \,|\, \text{CS miss}) = \mathbb{E}[m(D)]/(1 + \mathbb{E}[m(D)])$ is the conditional probability of a PIT hit given a CS miss, meaning the probability that the request is not the first request in the interval $[0, D)$ (the first request is forwarded). In other words, the forwarding probability equals the probability of a miss request times the probability that the request is the first request in the cycle (the next $\mathbb{E}[m(D)]$ requests on average are PIT hits).

4.1.1. Leveraging Residual Download Time Moments to Approximate Z and $\widehat{Z}$

In the above equations, R is a random variable distributed like the remaining download time of a sample request arriving at some time $t_i \in [0, D)$. Figure 1 depicts R_1, the remaining download time of the first request, $R_1 = D - t_1$. We proceed by first approximating the moments of R and then fitting a standard distribution using moment matching.

Let $r_n(t)$ be the sum of the n-th moments of the residual download times of all requests arriving in $[0, t)$ for some constant $t \leq D$: $r_n(t) = \mathbb{E}[\sum_{i=0}^{N_t}(t - t_i)^n]$, where the t_i are request times and N_t is the number of requests during interval $(0, t)$. To compute $r_n(t)$, we have,

$$r_n(t) = t^n + \int_0^t r_n(t - x)dF(x) = t^n + \int_0^t (t - x)^n dm(x),$$

where the second equality follows on rewriting the renewal equation in terms of the renewal function [28]. The moments of R satisfy

$$\mathbb{E}[R^n] = \frac{\mathbb{E}[r_n(D)]}{\mathbb{E}[m(D)] + 1}. \tag{14}$$

These moments can be used to derive a *phase type* distribution that fits the distribution of R arbitrarily closely [34]. In practice, in all our numerical evaluations, it has proved sufficient to fit just the first two moments; however we also observed that matching only the first moment is sufficient for IRM input. We will now rewrite expressions for Z and $\widehat{Z}$ as

$$Z = \int_0^\infty (F(r + T_C) - F(r))\, dR(r),$$

$$\widehat{Z} = \int_0^\infty \left(\widehat{F}(r + T_C) - \widehat{F}(r)\right) dR(r). \tag{15}$$

4.2. LRU CS with Filter

To improve CS hit rates we preferentially insert more popular contents, as identified by a filter placed in front of the CS. The filter consists of a double linked list of contents updated using the standard LRU policy on every *request* arrival. Filter performance can thus be derived using the classical LRU characteristic time approximation: the filter hit probability for content k of a filter of size M is $p_{hit}^{flt}(k) = F_k(T_M)$ where characteristic time T_M is such that $\sum_K \widehat{F}_k(T_M) = M$.

Filter size M should be set in such a way that $T_C \geq T_M$ which typically happens when $C \geq M$. For small values of M, the insertion probability will be very small and 2-LRU will behave optimally for IRM traffic (i.e., the cache will

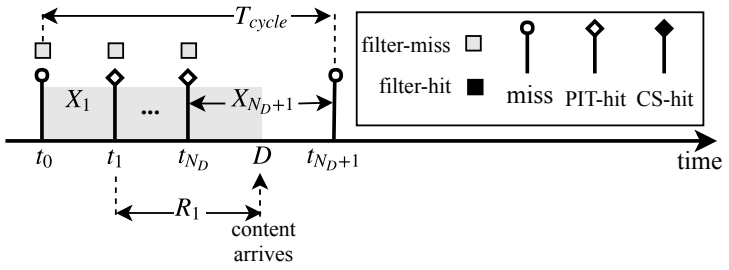


Figure 2: Request process and CS status for a given content under non-ZDD LRU with a filter.

store only the most popular contents) [4]. Optimality is no longer assured, however, when the request process exhibits temporal locality.

A content is inserted in the CS only if on download at least one of the requests in $[0, D)$ was a filter hit. Figure 2 illustrates the cycle between two forwarded requests (at t_0 and t_{N_D+1}) when the content is absent from the filter. This means the content is added to the filter at epochs t_i for $0 \le i \le N_D$ but always evicted before the next request at t_{i+1}.

To approximate p_{hit}^{cs} and p_{fwd}, we assume the filter and CS-PIT states are independent [4]. This independence assumption approximation was first proposed for k-LRU caches in [4] and shown there to be accurate in comparison to results of simulations. An exact evaluation without the independence assumption was provided in [4] for $k = 2$ and has since been extended for $k > 2$ by Gast and Van Houdt [30]. However, the accuracy of the independence assumption is confirmed in [30], especially for the present 2-LRU cache, and we adopt it here since it is considerably simpler to apply than the exact solution. Moreover, while the analysis could be extended to k-LRU caches for $k > 2$, we know from results in [4] that 2-LRU is more reactive and tends to have better performance when demand exhibits temporal locality.

Denoting the probability of insertion by q and applying the arguments of Sec. 4.1 above, we deduce,

$$p_{in}^{cs} = p_{hit}^{cs} \cdot \widehat{F}(T_C) + q \cdot (1 - p_{hit}^{cs}) \cdot \widehat{Z},$$
$$p_{hit}^{cs} = p_{hit}^{cs} \cdot F(T_C) + q \cdot (1 - p_{hit}^{cs}) \cdot Z,$$

where Z and $\widehat{Z}$ are given by (15). Solving these equations gives,

$$p_{in}^{cs} = q \cdot \frac{Z\widehat{F}(T_C) + \widehat{Z}(1 - F(T_C))}{1 - F(T_C) + Z \cdot q}, \tag{16}$$

$$p_{hit}^{cs} = \frac{Z \cdot q}{1 - F(T_C) + Z \cdot q}. \tag{17}$$

Note that these formulas would apply to any filter for which one can determine the insert probability q. They would apply in particular if q were simply a constant probability of insertion, as envisaged in the probabilistic cache policy [18, 4] or content specific, as in the access-time aware and utility maximization cache policy proposed, respectively, in [19] and [20].

For the present LRU filter, the requests illustrated in Figure 2 delimit independent cycles and $q = \mathbb{E}\left[1 - (1 - p_{hit}^{flt})^{N_D+1}\right]$. We approximate this as follows,

$$q \approx 1 - (1 - p_{hit}^{flt})^{[\mathbb{E}[m(D)]+1]}. \tag{18}$$

Recall that we have omitted dependence on the content k but in fact the above probabilities are all content specific. For k-LRU, the insertion probability q could be obtained using the same formula with p_{hit}^{flt} now relating to the last LRU filter and computed recursively using the ZDD k-LRU formulation given in [4, Eq. (11)]. Given expression (17) for the probability a given content is present in the CS, we can determine T_C numerically from (8) and thereby, the CS hit probabilities. The forwarding probability p_{fwd} is given by (13), as before.

5. Performance evaluation and insights

We first investigate the behavior of the CS-PIT system for a range of parameter settings under synthetic workloads, confirming the accuracy of our analysis by comparison with simulation results. We then use the analytical model to evaluate performance when contents have realistically long but finite lifetimes. Finally, to further validate the model and the derived insights, we perform some trace-based simulations.

5.1. System Configuration

We consider two instances of the request process: the Poisson process, corresponding to the IRM, and a process with 2-state hyper-exponential inter-request times: the inter-request interval for content k is drawn from an exponential distribution of rate $z\lambda_k$ with probability $z/(z+1)$ and an exponential distribution of rate λ_k/z with probability $1/(z+1)$, where z is a parameter that determines the degree of time locality in the request process. We set $z = 10$ to model strong correlation between requests. Observe that this process is also equivalent to an IPP where requests arrive at rate $\nu_n \approx 9 \cdot \lambda_n$ in an on-period and the off-period is almost 8 times longer than the on-period [27, Sec. 2.3]. We call this process 'hyper10'. Note that when $z = 1$, the resulting renewal process is a Poisson process and there is no time locality between requests. To validate the analysis of Sec. 4 and to investigate system behavior, we use the parameter settings in Table 1. We have used constant download delays, the same for all contents drawn from the range reported in [35]. The Zipf parameter α is set to 0.8 for all evaluations though we discuss the impact of alternative values in Sec. 5.5. Results are systematically presented for both LRU and 2-LRU policies.

5.2. Performance Impacts

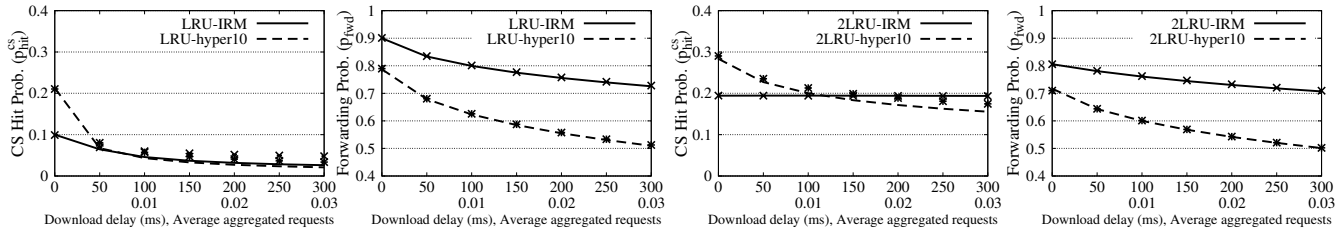
Results are displayed for two performance measures: the overall CS hit probability p_{hit}^{cs} and the overall forwarding probability p_{fwd}. The PIT hit probability can be derived from (1). Throughout this section, we depict the results for LRU CS with filter using the label 2-LRU. Simulation results are plotted as crosses. We have simulated 5 runs of 10^9 requests for each cross which ensures statistically stable results. The confidence interval is reported for each point, but the vertical error bars on each point is not larger than the thickness of the point and therefore not visible. The simulator is available on GitHub [6]. The plots, whose behavior is discussed below, confirm that the analytical model is generally very accurate. Note that 2-LRU is consistently better than LRU in all cases depicted in Figures 3 to 6. Similarly, time locality yields consistently higher hit probabilities and lower forwarding probabilities for hyper10 traffic compared to results for the IRM. We now comment on specific impacts revealed by each set of plots.

5.2.1. Download Delay

Figure 3 shows how performance depends on download delay D or more generally on the average number of requests aggregated during a download delay. The latter normalized variable, $\lambda D/K$, is shown as the second x axis in Figure 3. The CS hit probability decreases as D increases but this decrease is more than compensated by an increase in PIT hits yielding a decreasing trend for the forwarding probability for practical D values. Observe that for very small download delays, e.g., $0 < D < 0.1$ ms, the forwarding probability tends to increase slightly with D but this increase is too small to be visible in the figure. The effectiveness of the PIT is clearly higher for the longer delays and may therefore bring greater benefits in more remote areas of the Internet topology. These results show the PIT plays the role of a supplementary cache and can have a significant impact on performance. The difference in p_{fwd} between LRU and 2-LRU decreases as D increases suggesting the PIT compensates for the absence of filter.

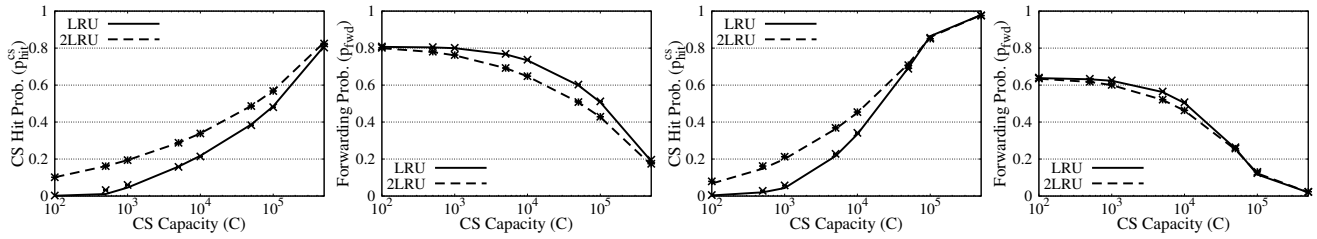
Table 1: Parameter Settings

Parameter	**Default**	**Range**
Zipf Parameter (α)	0.8	
Catalogue Size (K)	10^6	10^5 - 10^9
[Catalog Size/CS Capacity] Ratio (C/K)	10^{-3}	10^{-4} - 0.5
Download Delay (D)	100 ms	0 - 300 ms
Request Rate (λ rqt/s)	10^5	10 - 10^6
Filter Size (M)	C	



(a) LRU (b) 2LRU

Figure 3: The impact of download delay or average aggregated requests on CS hit probability and forwarding probability under LRU and 2LRU for fixed catalogue size $K = 10^6$, CS capacity $C = 1000$ and request rate $\lambda = 10^5$ (rqt/s). Average aggregated request values, $\lambda D/K$, are shown as the second x axis.



(a) IRM Traffic (b) Hyper10 Traffic

Figure 4: CS hit probability and forwarding probability versus CS capacity under non-ZDD LRU and non-ZDD 2LRU for fixed catalogue size $K = 10^6$ and request rate $\lambda = 10^5$ (rqt/s).

5.2.2. CS Capacity

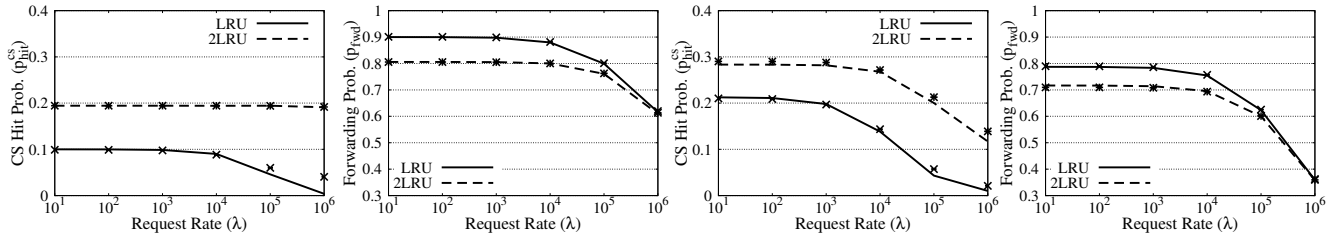
Figure 4 illustrates the impact of CS capacity. Note that the gain in CS hits of 2-LRU over LRU is especially significant for small caches where LRU is clearly inadequate. On the other hand, with the default download delay of 100 ms, the reduction in p_{hit}^{cs} is compensated by an increase in p_{hit}^{pit} so that both policies yield nearly the same forwarding rate, especially for hyper10 traffic. Temporal locality can thus bring an increase in PIT hit probability over that prevailing under IRM demand.

5.2.3. Traffic Intensity

It is well known that cache performance under the ZDD assumption is independent of traffic intensity in requests/sec since it depends only on the order of requests and not on their precise timing. This insensitivity is not preserved under the present non-ZDD model. Figure 5a for IRM input shows that the forwarding probability decreases significantly for high arrival rates thanks to an increasing probability of PIT hit. As the duration a pending request remains in the PIT is fixed at D the number of aggregated requests increases in proportion to λ. Similar trends are observed for the hyper10 request process. Temporal locality of requests, however, further accentuates the dependence of both CS and PIT performance on λ.

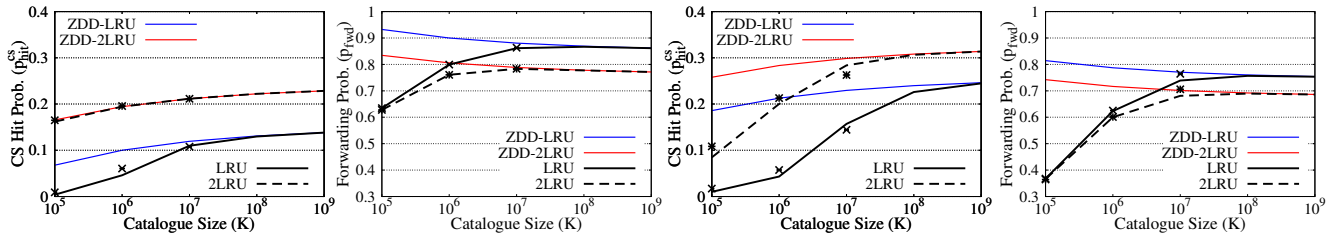
5.2.4. Catalogue Size

Figure 6 shows the impact of an increasing catalogue size. Results for very large catalogues are derived by analysis alone as simulation then becomes impractical. The catalogue size varies from 10^5 to 10^9 while other parameters have the default settings. The case for $D = 0$ (i.e., the usual ZDD assumption) is also shown for comparison. We observed in the previous analysis that high traffic densities can lead to a decrease of the CS hit probability as more requests miss the CS during the download time. In Figure 6, the per-content traffic intensity decreases as the catalogue size grows leading therefore to a CS hit probability that increases and tends to the ZDD value. Figure 6a suggests that the PIT hit probability under IRM input is negligible for large catalogues, i.e., under low per-content traffic intensity.



(a) IRM Traffic (b) Hyper10 Traffic

Figure 5: CS hit probability and forwarding probability versus request rate under non-ZDD LRU and non-ZDD 2LRU for fixed CS capacity $C = 10^3$ and catalogue size $K = 10^6$.



(a) IRM Traffic (b) Hyper10 Traffic

Figure 6: CS hit probability and forwarding probability versus catalogue size under LRU and 2-LRU (ZDD and non-ZDD) for fixed C/K ratio $= 10^{-3}$ and request rate $\lambda = 10^5$ (rqt/s).

For hyper10 traffic in Figure 6b, on the other hand, request time locality means PIT aggregation remains effective for bigger catalogues and the non-ZDD model is necessary to accurately predict performance.

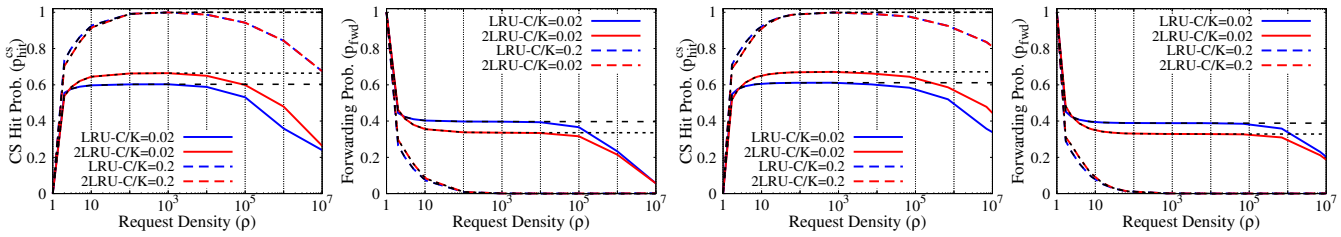
5.3. Impact of Finite Lifetime

We now complement the evaluation scenarios of the previous section using a more realistic model of popularity variation. The hyper-z model can artificially model temporal locality but hardly represents realistic variations since high activity periods, representing finite lifetimes, have the same mean number of requests so that content lifetime durations are inversely proportional to popularity. In this section we assume lifetimes have a given average duration. For illustration purposes, we set the same lifetime for all contents though the model would allow content specific durations.

Measurements reported in the literature show that the average lifetime of the most dynamic fraction of video on demand contents is around 2 days (see [24]). It is hardly practical to simulate a system with such a wide range of timescales, from sub-second download times to days long lifetimes, for moderate to high request densities. We therefore rely here on analytical results that allow us to explore a wider range of possible values. As explained in Sec. 3.4, we model time varying popularity using IPP renewal processes. The lifetime is identified with an exponential on-period of mean duration T_{on} while the exponential off-period is of mean duration $T_{off} = 9 \cdot T_{on}$. T_{off} must be large enough that it is considerably larger than the characteristic time T_C so that each on-period appears as a new content with respect to CS and PIT states.

We take parameter values from the paper [5] where the IPP model was first proposed. T_{on} and catalogue size K are set so that the rate at which 'new' contents occur, denoted γ, and the mean number of active contents are fixed. Thus $K = 10\gamma T_{on}$. We set $\gamma = 5 \cdot 10^4$ contents per day, the value reported in [5], and select T_{on} from 1, 7 and 30 days to explore a range of scenarios. Other parameters take default values from Table 1.

Figure 7 plots p_{hi}^{cs} and p_{fwd} for two relative cache sizes $C/K = .02$ and $C/K = .2$, as functions of a measure of request density, denoted ρ, for $T_{on} = 1$ day and $T_{on} = 7$ days. Request density is defined as the expected total number of requests occurring in a content lifetime. In the IPP model, there are "lifetimes" that in fact have no requests.



(a) T_{on} = 1 day (b) T_{on} = 7 days

Figure 7: CS hit probability and forwarding probability versus request density for different T_{on} durations under non-ZDD LRU and non-ZDD 2-LRU for two CS sizes $C = 0.02K$ and $C = 0.2K$. Black dashed lines represent the case for $D = 0$.

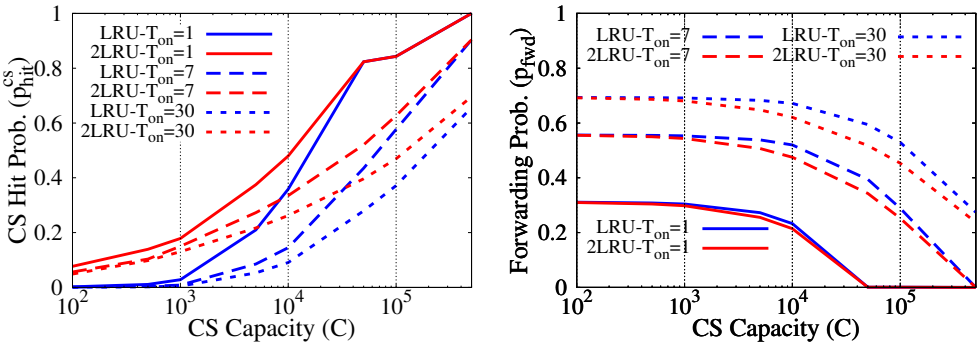


Figure 8: CS hit probability and forwarding probability versus CS capacity for different T_{on} durations under non-ZDD LRU and non-ZDD 2-LRU when request density is fixed to $\rho = 10^6$.

Considering actual lifetimes to be manifested by at least one request (as is the case in any trace of real traffic), we define ρ as follows,

$$\rho = \sum_{k=1}^{K}(1 + \nu_k T_{on})/K = 1 + \lambda/\gamma. \tag{19}$$

Cases (a) and (b) behave similarly for small ρ. The CS hit probability is small and, moreover, the 2-LRU policy can become less effective than LRU. This is explained by the low reactivity of these policies at low densities: the first request and first two requests in an on-period are necessarily misses for LRU and 2-LRU, respectively. The relative impact of these systematic misses is significant when $\rho < 10$. When C is smaller, the number of systematic misses relative to the misses due to content churn decreases and, as a result, we see a less steep line for $\rho < 10$ when $C/K = 0.02$ compared to $C/K = 0.2$.

As ρ increases, the CS hit rate attains a maximum that is flat over the range $10^2 \leq \rho \leq 10^4$ and roughly equal for both values of T_{ON}. This hit rate would be attained with IRM input with the same popularity distribution. The impact of systematic misses for the first and first 2 requests in a lifetime for LRU and 2-LRU, respectively, becomes negligible when the number of requests per lifetime is typically large. Moreover, the impact of PIT request aggregation is negligible until $\rho > 10^4$.

For very large densities, collapsed forwarding comes into play and the forwarding probability decreases significantly. The key variable is the average number of aggregated requests per download delay, as identified in Sec. 5.2.1, and equal here to $\rho D/T_{on}$. From Figure 3, the PIT significantly improves performance when this variable is greater than .01, roughly corresponding to $\rho > 10^4$ in case (a) and to $\rho > 10^5$ in case (b).

Figure 8 plots p_{hit}^{cs} and p_{fwd} as functions of CS capacity for different values of the average on-period duration. We set the request density to $\rho = 10^6$ in this scenario. The results again confirm that when ρ is adequately large, CS hit probability is roughly inversely proportional to the length of the on-period, i.e., content lifetime. On the other hand, for a fixed ρ value, the PIT is more effective when the ratio of download delay to lifetime is larger and/or the requests

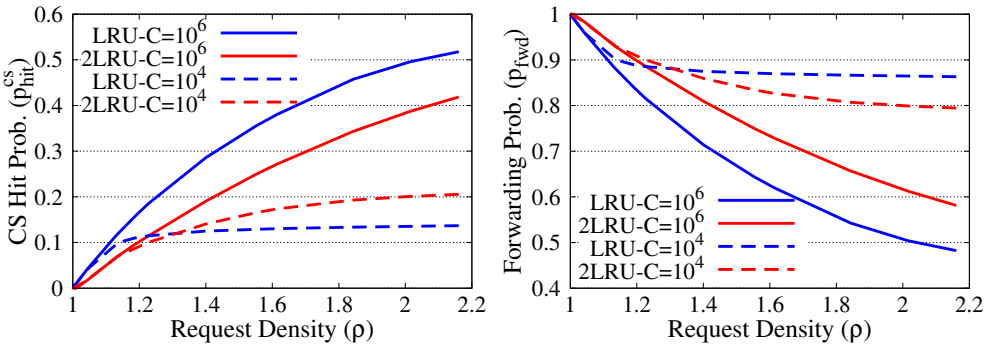


Figure 9: CS hit probability and forwarding probability versus request density under LRU and 2-LRU. The largest ρ value represents the results for the original trace.

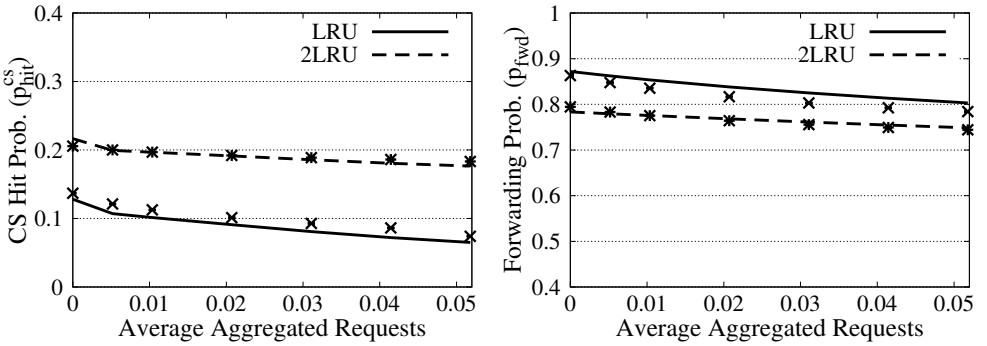


Figure 10: CS hit probability and forwarding probability versus download delay represented as average number of aggregated requests per download delay under LRU and 2LRU for CS capacity $C = 10^4$.

show more time locality, i.e., for $T_{on} = 1$.

5.4. Trace-based Evaluation

To further validate the analysis and the derived insights, we have performed trace-based simulations. We use a trace of YouTube video requests provided to us by the authors of [24]. This trace (labelled Trace 4 in [24]) contains 3.8M requests for 1.76M videos generated from 31124 distinct IP addresses within the network of an Italian ISP and was recorded over a 35 day period in 2012.

Excluding contents with less than 10 requests, lifetimes range from about 1 day for the most dynamic group of contents to about 25 days. Request density, defined above as the average number of requests during each content lifetime, is approximately $\rho \approx 2.16$. To simulate smaller densities we have to thin the observed request process: each request is retained with probability p and we simulate densities from $\rho \approx 1$ to $\rho \approx 2.16$ by ranging p between 0 and 1.

Figure 9 shows hit rates and forwarding probabilities for LRU and 2-LRU caches with capacities $C = 10^4$ and $C = 10^6$ as a function of ρ. The forwarding probability is just equal to the CS miss rate here for such low densities. The results confirm the analytical findings for small densities discussed in Sec. 5.3: the performance of both LRU and 2-LRU suffers from systematic misses and the less reactive 2-LRU is worse than LRU for the larger cache.

To study the larger request density regime where the PIT becomes effective in reducing the forwarding probability we must artificially increase the download delay. This is because the trace density of $\rho \approx 2.16$ is simply too low to produce request aggregation with a realistic download delay of 100 ms or less. Increasing the download delay allows us to explore performance over a range of average aggregated requests values.

The results are displayed in Figure 10. CS hit rates and forwarding probabilities are qualitatively similar to those shown in Figure 3 tending to validate the analysis. As further validation, Figure 10 plots trace simulation results together with analytical results derived by fitting the IPP model to the trace data. Specifically, we have used the data

provided in Table 2 in [24] to fit the mean on-time $T_{on}(k)$ and the request rates ν_k for the 5 lifetime classes identified in the table. The closeness of the analytical and simulation results confirms that the assumption of Poisson request arrivals within each lifetime is a reasonable approximation for the CS-PIT system, as previously noted in [24] for a simple cache. The small errors in model predictions are mainly due to significant bias in the lifetime estimates arising from the limited trace length and the small number of requests observed for many contents.

5.5. Discussion

The results of this section show that the effectiveness of PIT aggregation varies widely depending on the chosen scenario. It is useful therefore to discuss observed behavior in the light of known demand and network characteristics.

The PIT is very effective in reducing forwarding rates when demand is high and concentrated on a relatively small catalogue of contents (cf. Figure 5 and Figure 6). The key parameter is the average number of requests occurring in the download delay, $\lambda D/K$. For the considered Zipf(.8) popularity distribution, the PIT has a noticeable impact when $\lambda D/K > 0.01$. This condition may be satisfied for certain NDN deployments but perhaps not for core routers performing content retrieval from the entire Internet.

Known statistics on different types of content, like the Web or YouTube videos, suggest catalogues approaching the petabyte in total volume [36], while traces from real traffic observations reveal volumes of at least several tens of terabytes [21]. Converted to NDN chunks (close in size to IP packets) this suggests catalogues K in excess of 10^{10}. On the other hand, demand in a core NDN router with multiple 10 Gb/s links might generate $O(10^6)$ requests per second at peak times. The request rate per content item ($O(10^{-4})$) is still rather low for the PIT to be effective (in Figure 5, the same relative per content request rate would occur at $\lambda = 100$). The scenario would be more favorable with a more skewed popularity distribution (e.g., Zipf(1)) though most measurements in the field suggest this is not very likely (e.g., [21]).

The plots in Figures 3 to 6 demonstrate the generally positive impact on hit rate performance of time locality. On the other hand, Figure 7 and Figure 9 show that finite lifetimes can significantly reduce the effectiveness of reactive caching policies like LRU and 2-LRU when demand is relatively low. The critical parameter here is the expected number of requests in a content lifetime that we have termed request density, ρ.

To understand why performance is poor for low densities, note that an unlimited capacity LRU cache would have a hit rate of $(\rho - 1)/\rho$: for a trace like that used in Sec. 5.4, the total number of requests is ρ times the number of distinct contents while there is exactly 1 miss per content. This hit rate is only within 10% of the ideal hit rate of 1 (that would eventually be attained with IRM demand) for $\rho > 10$. The hit rate of an unlimited capacity 2-LRU cache would always be lower than that of LRU since only requests after the first two yield hits. For a limited cache, however, there is a crossover point as ρ increases. We have observed that the crossover occurs earlier for smaller caches since this point is where the selectivity of 2-LRU trumps the greater reactivity of LRU which happens earlier for smaller cache sizes, but it does not appear possible to explain this behaviour through simple closed-form formulas.

As a practical consequence, note that when density ρ is small, in an edge router delivering content from a large catalogue say, it may be necessary to perform proactive caching (i.e., to push the most popular contents to the CS) in order to significantly reduce the forwarding rate.

6. Conclusions

The characteristic time based analytical framework developed in this paper is both versatile and accurate. We have modelled the CS-PIT system with non-zero download delay applying LRU and 2-LRU cache replacement policies under general renewal request processes. Analytical results, whose accuracy is confirmed by simulations, enable an appraisal of the effectiveness of the PIT in reducing network traffic through the use of collapsed forwarding.

The effectiveness of the PIT naturally increases with the duration of the download delay. The more complex 2-LRU replacement policy gives higher CS hit rates than simple LRU in all cases but this advantage is mitigated in non-ZDD scenarios where the PIT is a meta cache that, like the filter, tends to improve the performance of the most popular contents.

The PIT is most effective when demand per content is relatively high such that several requests often occur in a download delay. This happens when overall demand is high, as in a core router, but only if this demand is not spread over a very large content catalogue. If demand is low, as in an access node or an enterprise router, and nevertheless spread over a large catalogue, the PIT is hardly effective in realizing collapsed forwarding.

When contents have a finite lifetime during which they are popular and receive requests (an approximate model of popularity variation), the above remarks on PIT effectiveness still apply. In addition we observed that both reactive cache policies, LRU and 2-LRU, can be ineffective when the expected number of requests per content is small. Whenever this case arises in practice it is preferable to implement a placement policy where the most popular contents are proactively pushed to the CS.

The above insights derived from the analytical model have been confirmed by the results of trace-driven simulation experiments.

Appendix A.

We compare the complexity and accuracy of the present analytical framework with that proposed by Dehghan *et al.* [3]. We first briefly review the formulas given in [3] for the performance of a non-ZDD LRU cache under general renewal process. We then discuss the complexity needed to compute different terms in the respective formulations and present the results of some numerical experiments.

In the notation of [3], Γ_t is the residual life (forward recurrence time) of the request process at time t after the start of a cycle. The distribution of Γ_t is [3, Eq. (7)],

$$\mathbb{P}(\Gamma_t \leq a) = F(t+a) - \int_0^t (1 - F(t+a-x))\, dm(x), \tag{A.1}$$

where $m(x)$ is the *renewal function* of a general renewal process and is given by 5.

The following expressions for p_{in}^{cs} and p_{hit}^{cs} are given in [3] (denoted o and h, respectively),

$$p_{in}^{cs} = \frac{\mathbb{E}[\Gamma_D] + \mathbb{E}[N]/\lambda - \mathbb{E}[\Gamma_{t_E}]}{\mathbb{E}[T_{cycle}]}, \tag{A.2}$$

$$p_{hit}^{cs} = \frac{\mathbb{E}[N]}{1 + \mathbb{E}[m(D)] + \mathbb{E}[N]}, \tag{A.3}$$

where N is the number of hits in a cycle and $\mathbb{E}[T_{cycle}]$ is the expected cycle length. Γ_D and Γ_{t_E} are the residual life of the request process at content insertion time D and at content eviction time t_E, respectively.

In the notation of Figure1, $T_{cycle} = \sum_{i=1}^{1+N_D+N} X_i$ where N_D is the number of PIT hits ($\mathbb{E}[N_D] = m(D)$). The upper summation limit is a stopping time for the sequence X_i and consequently we can apply Wald's equation [3, Eq. (10)],

$$\mathbb{E}[T_{cycle}] = (1 + \mathbb{E}[m(D)] + \mathbb{E}[N])/\lambda. \tag{A.4}$$

The mean number of hits $\mathbb{E}[N]$ is given by [3, Eq. (8)] which, for constant D and characteristic time T_C reads,

$$\mathbb{E}[N] = \frac{\mathbb{P}(\Gamma_D \leq T_C)}{1 - F(T_C)}. \tag{A.5}$$

To compute T_C using the expression for p_{in}^{cs} and fixed point equation (8), $\mathbb{E}[N]$, $\mathbb{E}[\Gamma_D]$ and $\mathbb{E}[\Gamma_{t_E}]$ must be computed at each iteration. $\mathbb{E}[N]$ can be computed simply using (A.5). We now explain the complexity of computing $\mathbb{E}[\Gamma_D]$ and $\mathbb{E}[\Gamma_{T_E}]$. Note that no method is proposed in [3] for computing these terms. We therefore use our own method to compute them.

To compute the difference $\mathbb{E}[\Gamma_D] - \mathbb{E}[\Gamma_{t_E}]$ in (A.2) we distinguish the cases where $\Gamma_D > T_C$, and there are no hits in the cycle (Figure1a), and $\Gamma_D \leq T_C$ with at least one hit (Figure1b). In the first case, $\Gamma_D - \Gamma_{t_E} = T_C$. In the second, $\mathbb{E}[\Gamma_D] = \mathbb{E}[\Gamma_D | \Gamma_D \leq T_C]$ and $\mathbb{E}[\Gamma_{t_E}] = \mathbb{E}[(X - T_C) | X > T_C]$. We deduce,

$$\mathbb{E}[\Gamma_D] - \mathbb{E}[\Gamma_{t_E}] = \mathbb{P}(\Gamma_D > T_C) \cdot T_C + \mathbb{P}(\Gamma_D \leq T_C) \cdot \Big(\frac{\mathbb{E}[\Gamma_D \cdot 1_{\{\Gamma_D \leq T_C\}}]}{\mathbb{P}(\Gamma_D \leq T_C)} - \frac{\mathbb{E}[(X - T_C) \cdot 1_{\{X > T_C\}}]}{\mathbb{P}(X > T_C)} \Big), \tag{A.6}$$

where

$$\mathbb{E}[\Gamma_D \cdot 1_{\{\Gamma_D \leq T_C\}}] = \int_0^{T_C} \big(\mathbb{P}(\Gamma_D \leq T_C) - \mathbb{P}(\Gamma_D \leq t) \big) dt, \tag{A.7}$$

$$\mathbb{E}[(X - T_C) \cdot 1_{\{X > T_C\}}] = \int_{T_C}^{\infty} \mathbb{P}(X > x) dx. \tag{A.8}$$

Computing (A.7) for every iteration over T_C is the main source of complexity. It proves significantly more complex than computing (15) in the approach proposed in this paper due to larger effective domain and the complexity of computing (A.1). To illustrate this difference we compare the execution time and accuracy of the two approaches for some reference scenarios. We refer to our approach as 'method-A' and that proposed in [3] as 'method-B'.

We implement both methods in Python using the built-in integration function `scipy.integrate.quad` to compute the integrals (available on GitHub [6]). We run both methods on the same server, repeating each experiment 10 times and reporting the minimum execution time to decrease the impact of uncontrolled tasks running on the server. The iterative algorithm used to solve the fixed point equation (15) is the same for both methods. We start from $T_C = C/\lambda$ and continue until $|C - \sum_{k=1}^{K} p_{in}^{cs}(k)| > \epsilon$ for $\epsilon = 10^{-3}$. We consider the hyper10 request process defined in Sec. 5.1 and evaluate the overall CS hit probability for different catalogue sizes. Other parameters are the default values reported in Table 1.

Table A.2 compares the execution times for two catalogue sizes, 10^6 and 10^7. The complexity of method-B leads to a significantly longer execution time that becomes prohibitive for large catalogues.

Table A.2: Comparison of method-A and method-B for different catalogue sizes under non-ZDD LRU for fixed C/K ratio = 10^{-3} and request rate $\lambda = 10^5$ (rqt/s) and $D = 100$ ms.

Catalogue Size (K)	**Method**	**Execution Time**
10^6	method-A	1h 15 min
	method-B	5h 9 min
10^7	method-A	3h 38 min
	method-B	32h 17 min

Acknowledgments

The work presented in this article has benefited from the support of NewNet@Paris, Cisco's Chair "Networks for the Future" at Telecom ParisTech (`http://newnet.telecom-paristech.fr`). Any opinions, findings or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of partners of the Chair.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, Networking named content, in: Proceedings of the 5th international conference on Emerging networking experiments and technologies, ACM, 2009, pp. 1–12.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, et al., Named data networking, ACM SIGCOMM Computer Communication Review 44 (2014) 66–73.

[3] M. Dehghan, B. Jiang, A. Dabirmoghaddam, D. Towsley, On the analysis of caches with pending interest tables, in: Proceedings of the 2nd ACM Conference on Information-Centric Networking, ACM, 2015, pp. 69–78.

[4] M. Garetto, E. Leonardi, V. Martina, A unified approach to the performance analysis of caching systems, ACM Transactions on Modeling and Performance Evaluation of Computing Systems 1 (2016) 12:1–12:28.

[5] M. Garetto, E. Leonardi, S. Traverso, Efficient analysis of caching strategies under dynamic content popularity, in: Proceedings of INFOCOM, IEEE, 2015, pp. 2263–2271.

[6] Code for simulation and analysis, [Online; accessed Oct., 18, 2019], Available: `https://github.com/mahdieh88/cache-pit-sim.git`.

[7] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, G. Caire, The role of caching in future communication systems and networks, IEEE Journal on Selected Areas in Communications 36 (2018) 1111–1125.

[8] R. Fagin, Asymptotic miss ratios over independent references, Journal of Computer and System Sciences 14 (1977) 222–250.

[9] C. Fricker, P. Robert, J. Roberts, A versatile and accurate approximation for lru cache performance, in: Proceedings of ITC 24, IEEE, 2012, pp. 1–8.

[10] H. Dai, B. Liu, H. Yuan, P. Crowley, J. Lu, Analysis of tandem pit and cs with non-zero download delay, in: Proceedings of INFOCOM, IEEE, 2017, pp. 1–9.

[11] P. R. Jelenković, A. Radovanović, The persistent-access-caching algorithm, Random Structures & Algorithms 33 (2008) 219–251.

[12] B. M. Maggs, R. K. Sitaraman, Algorithmic nuggets in content delivery, ACM SIGCOMM Computer Communication Review 45 (2015) 52–66.

[13] N. Carlsson, D. Eager, Worst-case bounds and optimized cache on mth request cache insertion policies under elastic conditions, Performance Evaluation 127 (2018) 70–92.

[14] P. Jelenkovic, X. Kang, A. Radovanovic, Near optimality of the discrete persistent access caching algorithm, in: International Conference on Analysis of Algorithms, pp. 201–222.
[15] S.-E. Elayoubi, J. Roberts, Performance and cost effectiveness of caching in mobile access networks, in: Proceedings of the 2nd ACM Conference on Information-Centric Networking, ACM, 2015, pp. 79–88.
[16] N. Gast, B. Van Houdt, Transient and steady-state regime of a family of list-based cache replacement algorithms, SIGMETRICS Performance Evaluation Review 43 (2015) 123–136.
[17] G. Casale, Analyzing replacement policies in list-based caches with non-uniform access costs, in: Proceedings of INFOCOM, IEEE, 2018, pp. 432–440.
[18] I. Psaras, W. K. Chai, G. Pavlou, Probabilistic in-network caching for information-centric networks, in: Proceedings of the 2nd Edition of the ICN Workshop on Information-Centric Networking, ACM, 2012, pp. 55–60.
[19] G. Neglia, D. Carra, M. Feng, V. Janardhan, P. Michiardi, D. Tsigkari, Access-time-aware cache algorithms, ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS) 2 (2017) 21:1–21:29.
[20] G. Neglia, D. Carra, P. Michiardi, Cache policies for linear utility maximization, IEEE/ACM Transactions on Networking 26 (2018) 302–313.
[21] C. Imbrenda, L. Muscariello, D. Rossi, Analyzing cacheable traffic in isp access networks for micro cdn applications via content-centric networking, in: Proceedings of the 1st ACM Conference on Information-Centric Networking, ACM, 2014, pp. 57–66.
[22] N. Carlsson, D. Eager, Ephemeral content popularity at the edge and implications for on-demand caching, IEEE Transactions on Parallel and Distributed Systems 28 (2017) 1621–1634.
[23] F. Olmos, B. Kauffmann, An inverse problem approach for content popularity estimation, in: Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools, ICST, 2016, pp. 33–40.
[24] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, S. Niccolini, Temporal locality in today's content caching: why it matters and how to model it, ACM SIGCOMM Computer Communication Review 43 (2013) 5–12.
[25] F. Olmos, B. Kauffmann, A. Simonian, Y. Carlinet, Catalog dynamics: Impact of content publishing and perishing on the performance of a LRU cache, in: Proceedings of ITC 26, IEEE, 2014, pp. 1–9.
[26] E. Leonardi, G. L. Torrisi, Least recently used caches under the shot noise model, in: Proceedings of INFOCOM, IEEE, 2015, pp. 2281–2289.
[27] W. Fischer, K. Meier-Hellstern, The markov-modulated poisson process (mmpp) cookbook, Elsevier Performance Evaluation 18 (1993) 149–171.
[28] S. Karlin, H. M. Taylor, A first course in Stochastic Processes, 2 ed. Academic Press, 1975.
[29] H. Che, Y. Tung, Z. Wang, Hierarchical web caching systems: Modeling, design and experimental results, IEEE Journal on Selected Areas in Communications 20 (2002) 1305–1314.
[30] N. Gast, B. Van Houdt, Asymptotically exact ttl-approximations of the cache replacement algorithms lru (m) and h-lru, in: Proceedings of ITC 28, IEEE, 2016, pp. 157–165.
[31] B. Jiang, P. Nain, D. Towsley, On the convergence of the ttl approximation for an lru cache under independent stationary request processes, ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS) 3 (2018) 20.
[32] S. Basu, A. Sundarrajan, J. Ghaderi, S. Shakkottai, R. Sitaraman, Adaptive ttl-based caching for content delivery, IEEE/ACM transactions on networking 26 (2018) 1063–1077.
[33] N. C. Fofack, M. Dehghan, D. Towsley, M. Badov, D. L. Goeckel, On the performance of general cache networks, in: Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools, ICST, 2014, pp. 106–113.
[34] W. Whitt, Approximating a point process by a renewal process, i: Two basic methods, Operations Research 30 (1982) 125–147.
[35] R. Singh, A. Dunna, P. Gill, Characterizing the deployment and performance of multi-cdns, in: Proceedings of the Internet Measurement Conference, ACM, 2018, pp. 168–174.
[36] C. Fricker, P. Robert, J. Roberts, N. Sbihi, Impact of traffic mix on caching performance in a content-centric network, in: NOMEN Workshop, 2012, pp. 310–315.